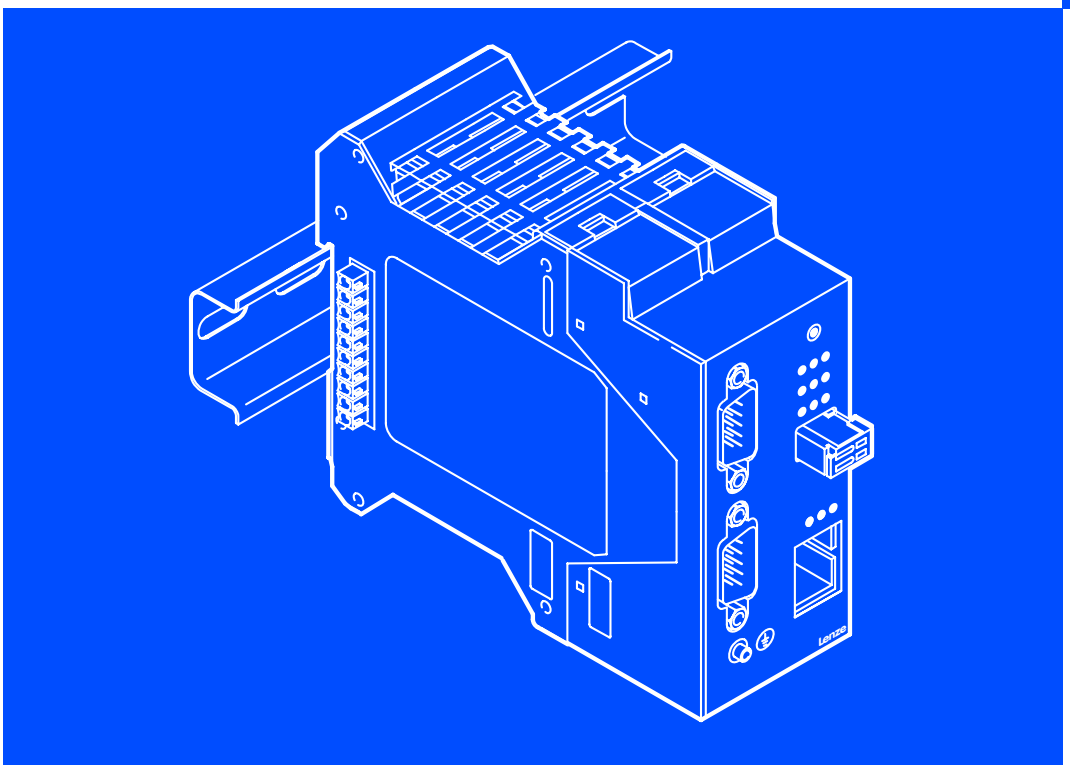
EDSTCXN
13145612

Software manual

ETC Motion Control



ETCxCxxx

Operating System NC

Lenze

Tip!

Current documentation and software updates concerning Lenze products can be found on the Internet in the ”Services & Downloads” area under

http://www.Lenze.com

© 2006 Lenze Drive Systems GmbH, Hans-Lenze-Straße 1, D-31855 Aerzen
No part of this documentation may be reproduced or made accessible to third parties without written consent by Lenze Drive Systems GmbH.
All information given in this documentation has been selected carefully and complies with the hardware and software described. Nevertheless, deviations cannot be ruled out. We do not take any responsibility or liability for damages which might possibly occur. Necessary corrections will be included in subsequent editions.

1 Preface and general information 10

1.1 About this Manual 10

1.2 For which products is the manual valid? 11

1.3 Legal regulations 12

2 Getting started 13

2.1 System overview 13

2.1.1 Examples for an automation system 13

2.1.2 Layout example for an ETC island 15

2.1.3 Connecting ETCHx and PC 16

2.2 Status display 17

2.3 Commissioning steps (overview) 20

2.4 Establishing the communication between PC and ETCHx 21

2.4.1 Starting ETCHx 21

2.4.2 Starting the terminal program ”HyperTerminal” and activating the monitor interface 21

2.4.3 Setting the operating mode of the ETCHx 25

2.4.4 Assigning the IP address of the ETCHx 28

2.5 Configuring ETC-MMI and ETC-MMI gateway 30

2.6 Installing ETC-MMI 31

2.6.1 Building in and installing the ETCPx 31

2.7 Starting ETC-MMI 32

2.7.1 Switching the language in the ETC-MMI 33

2.7.2 Establishing a connection between ETC-MMI and ETC 34

2.8 Parameterising drives via machine constants 36

2.8.1 Overview of the most important machine constants 36

2.8.2 Machine constant file ETCxC.mk 39

2.8.3 Notes on loading the MK file into the control system 40

2.8.4 Example for adapting a machine constant file 41

2.8.5 Adapting machine constants in the ETC-MMI 42

2.8.6 Checking the parameters of the drives 43

2.8.7 Testing the drives in inching mode 44

2.9 CNC programming according to DIN 66025 45

2.9.1 G-functions 45

2.9.2 M-functions 46

2.10 Creating a CNC sample program 47

2.10.1 Calling the text editor in the ETC-MMI 47

2.10.2 Entering and saving a CNC program 48

2.10.3 Loading the CNC program into the control system and starting it 49

2.10.4 Extending the CNC program 50

2.11 ETC PLC programming with CoDeSys 51
2.11.1 Installing CoDeSys 51
2.11.2 Configuring the control system in the ETC-CoDeSys 51
2.12 Creating a PLC sample program 60
2.12.1 Required hardware 60
2.12.2 Starting and configuring the PLC sample program 61
2.12.3 Loading the PLC sample program into the control system 68
2.13 Testing CNC and PLC program 72
2.14 PLC keys in the ETC-MMI 75
2.14.1 Labelling of the PLC keys in the ETC-MMI 75
2.14.2 Calling the signals in the PLC 76
2.14.3 Function of the keys 76
2.15 Operation via a Lenze-HMI 77
2.15.1 Settings for the connection of a Lenze-HMI H505 77
2.15.2 Functional description HMI505 operation 81
2.16 Updating the firmware of the ETCHx in the ”Standalone” operating mode 83
2.16.1 Calling the boot monitor in the control system 83
2.16.2 Querying the version of the firmware 84
2.16.3 Updating the firmware 84

3 CNC programming 87

3.1 Basics 87
3.2 G functions 91
3.2.1 Overview of G functions 91
3.2.2 G functions individual descriptions 94
3.3 Formula processor 172
3.3.1 Arithmetic operations 172
3.4 Block extensions 174
3.4.1 Parameter assignment P 174
3.4.2 H functions 175
3.4.3 M functions 175
3.4.4 Q functions 177
3.4.5 S functions 178
3.4.6 T functions 179
3.5 Data fields 180
3.5.1 P field 180
3.5.2 Q field 189

4 Machine constants 190

4.1 Basics 190

4.2 Test settings 192

4.2.1 MK_TEST_OHNEMECHANIK 192

4.2.2 MK_SPS_DUMMY 192

4.3 Software configuration 193

4.3.1 MK_KUNDE 193

4.3.2 MK_VOREINSTELLUNG 193

4.3.3 MK_NCPROG_OHNE_KOMMENTARE 194

4.3.4 MK_NCPROG_NICHT_INS_EEPROM 194

4.3.5 MK_METRISCH 194

4.3.6 MK_CONST_REL_MM 195

4.3.7 MK_CONST_REL_INCH 195

4.3.8 MK_KONTURFEHLER 195

4.3.9 MK_LAH_GRENZWINKEL 196

4.3.10 MK_RADIUS_B_BEWERTUNG 196

4.3.11 MK_EPSILONMM 197

4.3.12 MK_EPSILONGRAD 197

4.3.13 MK_OVERRIDEMAX 197

4.3.14 MK_FEHLERRESTART 197

4.3.15 MK_S0T0_VERSATZ_ERLAUBT 198

4.3.16 MK_S_VERSATZSPERRE 198

4.3.17 MK_CANOPEN_BAUDRATE 198

4.3.18 MK_DELTAT 199

4.4 Storage space reservation 200

4.4.1 MK_SPS_SPEICHERGROESSE 200

4.4.2 MK_SPS_DATENGROESSE 200

4.4.3 MK_SPV_SPEICHERGROESSE 200

4.4.4 MK_SPV_SYMBOLANZAHL 201

4.4.5 MK_HEADERANZAHL 201

4.4.6 MK_KANALANZAHL 201

4.4.7 MK_LAH_VORLAUFTIEFE 202

4.4.8 MK_LAH_RUECKLAUFGRENZE 202

4.4.9 MK_PFELD_GROESSE 203

4.5 Configuration of axes - Basics 204

4.5.1 MK_CANDRIVES 204

4.5.2 MK_APPLACHSIDX 205

4.6 Configuration of axes - Assignment and evaluation 207

4.6.1 MK_CANDRIVES 207

4.6.2 MK_APPLACHSIDX 207

4.6.3 MK_ACHSENART 208

4.7 Configuration of axes - Resolution 209
4.7.1 MK_IMPULSE 209
4.7.2 MK_WEG 209
4.7.3 MK_MASSSTAB 209
4.8 Configuration of axes - Operating range 210
4.8.1 MK_GRUNDOFFSET 210
4.8.2 MK_SW_ENDS_MINUS, MK_SW_ENDS_PLUS 210
4.8.3 MK_SW_ENDS_MIT_RAMPE 210
4.8.4 MK_SCHLEPPGENAUHALT 210
4.8.5 MK_GENAUHALTZEIT 210
4.9 Configuration of axes - Controller settings 211
4.9.1 MK_T2 211
4.10 Configuration of axes - Referencing 212
4.10.1 MK_REF_RICHTUNG_UND_FOLGE 212
4.11 Configuration of axes - speed and acceleration 213
4.11.1 MK_MODVMAX 213
4.11.2 MK_VMAX 213
4.11.3 MK_BESCHL, MK_BREMS 213
4.11.4 MK_T_BESCHL 213
4.11.5 MK_VBAHNMAX 214
4.11.6 MK_BAHNBESCHL, MK_BAHNBREMS 214
4.11.7 MK_T_BAHNBESCHL 214
4.12 Configuration of axes - Correction of axes 215
4.12.1 MK_SPINDELUMKEHRSPIEL 215
4.13 Configuration of axes - Handwheels 218
4.13.1 MK_CANDRIVES 218
4.13.2 MK_APPLACHSIDX 218
4.13.3 MK_ACHSENART 218
4.13.4 MK_IMPULSE 218
4.13.5 MK_WEG 218
4.13.6 MK_HANDRADZUORDNUNG 218
4.13.7 MK_HANDRADFAKTOR 219
4.13.8 MK_HANDRADFILTER 219
4.14 Configuration of axes - Synchronous axes 220
4.14.1 MK_ACHSENART 220
4.14.2 MK_SYNCHRONABWEICHUNG 220
4.14.3 MK_SYNCHRONOFFSET 220

4.15 Technology-specific settings 221
4.15.1 MK_MFKT_UPR_TABELLE 221
4.15.2 MK_TECHNOLOGIEDATEN1 ... MK_TECHNOLOGIEDATEN4 . . . 221
4.15.3 MK_MASCH_POLAR_KART 222
4.15.4 MK_KARTESISCH_ACHSNR 222
4.15.5 MK_POLAR_ACHSNR 223
4.15.6 MK_WLK_C_GRENZWINKEL 223
4.15.7 MK_WLK_C_OFFSET 223
4.15.8 MK_WLK_VERWEILZEIT 224
4.15.9 MK_X_WINKEL 224
4.15.10 MK_GEWINDE_VMAX 224
4.15.11 MK_DW224_255 225
4.16 List of machine constants 226

5 Interface PLC <-> NC operating system 240
5.1 Definitions 240
5.1.1 Data block 0 242
5.1.2 Data block 1 245
5.1.3 Data block 2 268
5.2 Extended interface for MMI functions 270
5.2.1 Data blocks 8 ... 14 270
5.2.2 Data block 15 273

6 ET -MMI gateway 276
6.1 Installing the ETC-MMI gateway 276
6.2 Starting the ETC-MMI gateway 277
6.3 Configuring the ETC-MMI gateway 278
6.3.1 Connection - Setting up connections 278
6.3.2 Trace – Error logbook 281
6.3.3 About – Version information 282
6.4 Mmigtway.ini 283
6.4.1 Example of the file ”mmigtway.ini” 284
6.5 Communication values in the DPR area 285

7 ETC-MMI 286

7.1 Installing ETC-MMI 286

7.2 Starting ETC-MMI 288

7.3 Operating ETC-MMI 289

7.3.1 Display elements of the program interface 289

7.3.2 Operational controls of the program interface 290

7.3.3 Help function 291

7.3.4 Configuration file 291

7.3.5 Language switch 291

7.3.6 Passwords 292

7.3.7 Notes, warnings, error messages 292

7.4 ”Setup” operating mode 293

7.5 ”Automatic” operating mode 298

7.6 ”Programming” operating mode 302

7.6.1 ASCI editor 305

7.6.2 File manager 308

7.6.3 Cycle programming 310

7.7 ”Diagnostics” operating mode 312

7.8 Appendix 321

7.8.1 Language file (SPRACHE.TXT) 321

7.8.2 Cycle programming 325

7.8.3 Configuration file (DELPHMMI.INI) 327

8 PLC programming 336

8.1 ETC PLC programming with CoDeSys 336

8.2 CoDeSys installation 337

8.2.1 System requirements for CoDeSys V2.xx 337

8.2.2 Installing software 337

8.3 Connecting ETC and PC 338

8.3.1 V.24 Interface 338

8.3.2 Ethernet interface (only ETCHx, DIN rail design) 338

8.3.3 DPR interface (only ETCPx, PCI insert card) 339

8.4 Project planning 340

8.4.1 Target system setup 340

8.4.2 Configuring PLC tasks of the ETCxM 340

8.4.3 Configuring PLC tasks of the ETCxC 342

8.4.4 Configuring I/O modules 343

8.4.5 Addressing 349

8.4.6 Remanent variables 350

8.4.7 Object directory (parameter manager) 350

8.5 Network variables 353
8.5.1 Settings in the target system 353
8.5.2 Settings in the global variable list 354
8.6 Generate program 355
8.7 Interface to the ETC 356
8.7.1 Data blocks 356
8.7.2 System variables of the ETCxC 356
8.7.3 System variables of the ETCxM 359
8.7.4 Using machine constants in the ETCxC 359
8.7.5 Using machine constants in the ETCxM 359
8.7.6 Parameter field of the ETCxC 359
8.7.7 Operating data of the ETCxC 360
8.7.8 Reading error messages for the ETCxM 360
8.8 Library 361
8.8.1 General functions 361
8.8.2 V24 functions 377
8.8.3 FILE IO functions 383
8.8.4 Memory access functions 393
8.8.5 CANopen functions 398
8.8.6 CAN functions (only ETCxM) 410
8.8.7 MMI communication functions 414
8.8.8 Realtime clock (only ETCxM) 417
8.9 Library ServerSDO.lib 418
8.9.1 InitServerSdo 418

9 Index 419

1 Preface and general information

1.1 About this Manual

Target group

This manual is intended for persons who program and commission the ETC Motion Control System under the ”NC” operating system.

Contents

The software manual ”NC Operating System” contains information on the following topics:

- ► Getting started – a chronological description of the commissioning steps
- ► CNC programming – working with G functions, formula processor, block extensions and data fields
- ► Machine constants – parameterisation of drives
- ► PLC-ETC interface – data blocks and their functions
- ► ETC-MMI gateway – communication between Windows applications and the ETC
- ► ETC-MMI – integrated development environment for the creation of CNC programs
- ► CoDeSys – integrated development environment for the creation of PLC programs

Further information

The ETC Hardware Manual contains information on the following topics:

- ► Technical data
- ► Structure and function of the system components including interface description
- ► Mounting, connecting and maintaining system components

1.2 For which products is the manual valid?

Standard device

①					
ETC	xx	0	xx	1A	10

Product

Version
HC = DIN rail, CNC core
PC = PCI plug-in card, CNC core

Number of axes
02 = 2 axes
04 = 4 axes
08 = 8 axes
12 = 12 axes

Hardware version

Software version

ETCHC0xx

ETCPC0xx

Modules

①			
ETCH	xxxx	1A	10

Product

N003 = power supply unit
T000 = bus termination module
I008 = 8 dig. inputs
I016 = 16 dig. inputs
U008 = 8 dig. outputs
U016 = 16 dig. outputs
A022 = 2 analogue inputs and outputs each

Hardware version

Software version

ETCHxxxx

1.3 Legal regulations

Marking

The components of the ETC Motion Control System are clearly marked by the contents of the nameplate.

Manufacturer

Lenze Drive Systems GmbH, Postfach 101352, D-31763 Hameln

CE conformity

Compliant with EC Directive ”Electromagnetic compatibility”

Application as intended

Components of the ETC Motion Control System

- ► must only be operated under the operating conditions described in the ETC Hardware Manual.
- ► are not approved for the use in explosive environments.
- ► comply with the protection requirements of the EC Directive ”Low voltage”.
- ► are no machines in the sense of the EC Directive ”Machines”.
- ► are no household appliances, as components they are intended for industrial use only.

The downstream user is responsible for ensuring that the EC Directives are complied with in machine use.

Any other use shall be deemed inappropriate!

Liability

The information, data and notes in this manual were up to date at the time of printing. No claims for the modification of systems and components that have already been supplied may be made on the basis of the specifications, illustrations and descriptions in this manual.

No liability is accepted by Lenze as to the suitability of any of the procedures or circuit recommendations included here.

The information in this manual describe the properties of the products without guaranteeing them. No liability will be accepted for damage or disturbance caused by:

- ► ignoring this manual
- ► unauthorised alterations to the components of the ETC Motion Control System
- ► operating errors and incorrect working on or with the ETC Motion Control System

Warranty

See terms of sales and delivery of Lenze Drive Systems GmbH.

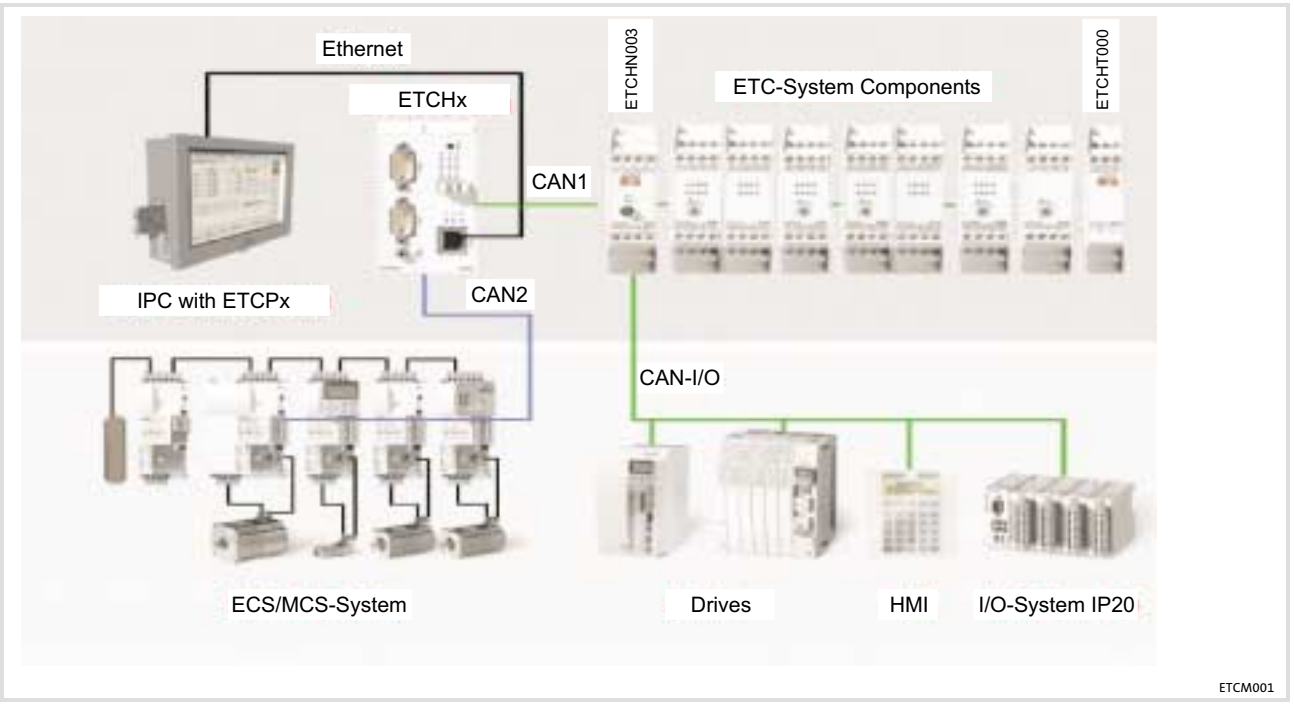
Report any claims under warranty to Lenze immediately on discovery of the defect or fault. The warranty is void in all cases where liability cannot be established.

2 Getting started

This chapter explains the basics of the ETC system and describes the procedure for realising an automation task.

2.1 System overview

2.1.1 Examples for an automation system



ETCHx / ETCPx

The core of the automation system is the ETC control in the top hat rail design (ETCHx) or as PCI insert card (ETCPx).

The top hat rail design ETCHx is normally used in a so-called ETC island (📖 15); it communicates via Ethernet with the applications on the IPC (or standard PC).

The ETCPC is inserted into the IPC (or standard PC) and communicates with the applications via the PCI bus.

Both designs have two separate CAN busses:

- ► At CAN1 bus (or also ME bus) the I/O modules and any operating components are connected. Third party devices must comply with teh DS401 profile of the CANopen specification.
- ► The drives (e.g. the ECS compact servo) are connected to the connection for the Motion CAN bus (CAN2) at the front plate. External drives must comply with the profile DS402 of the CANopen specification and in particular support the ”Interpolated Position Mode”.

To carry out its allocated control function the ETC control needs various programs which are transferred from the IPC (or standard PC):

- Operating system or firmware of the control (e.g. ETCHC.rsc)
- PLC programs (e.g. SPSDummy.prg)
- CNC programs; i.e. cycle and DIN programs (e.g. 9000.zyk or Nikolaus.din)

IPC

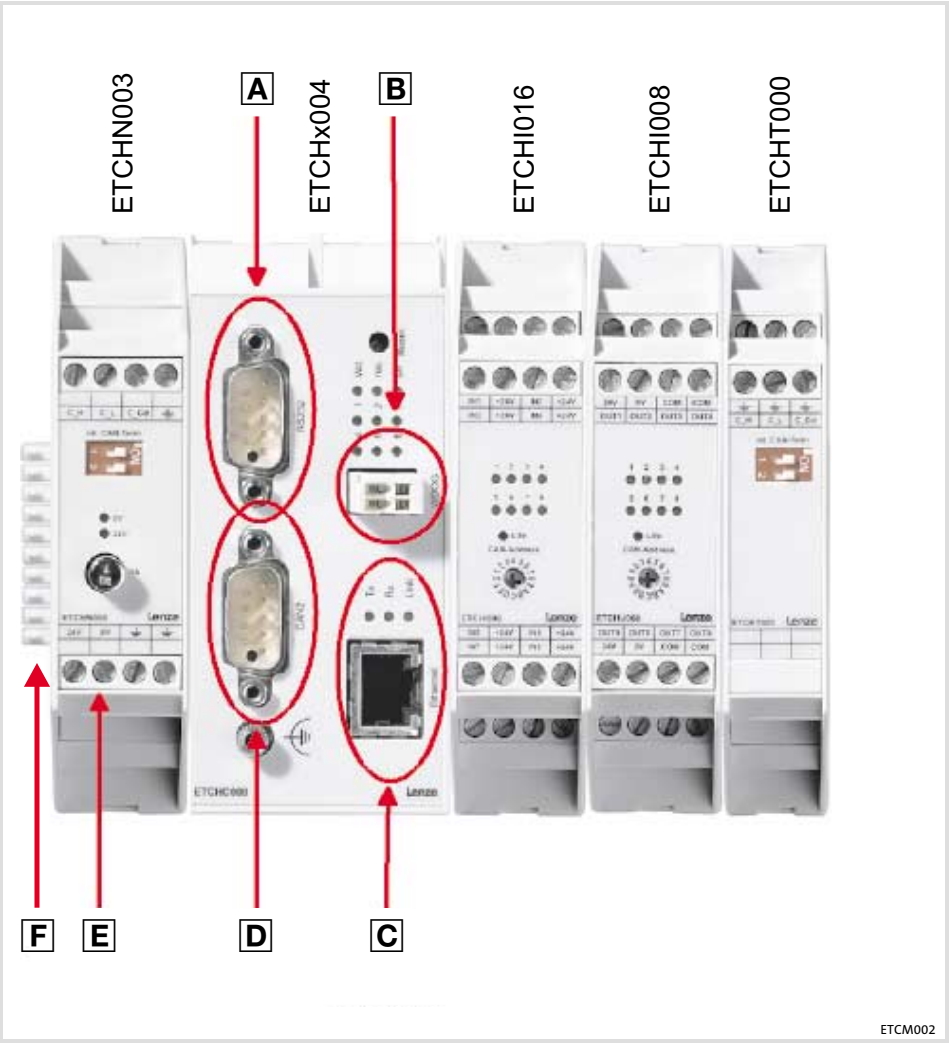
The ETC control is operated and maintained via the IPC (or standard PC). The following applications might run on it:

- Windows 2000 or XPoperating system
- Terminal program (e.g. HyperTerminal) for establishing the Ethernet connection between IPC and ETCHx control and for the configuration of the ETCHx control via the monitor interface (e.g. firmware update). This is not required for the ETCPx.
- ETC-MMI gateway as communications program between Windows applications and the ETC control.
- Lenze ETC-MMI for the configuration, operation and monitoring of the ETC control and for creating CNC programs.
- CoDeSys development environment for the creation and testing of PLC programs.

Note!

An external keyboard is required at the IPC for commissioning. It is not required for normal operation.

2.1.2 Layout example for an ETC island



A Serial interface (RS232)
B Watchdog (e.g. for emergency stop chain)
C Ethernet
D Motion CAN bus (CAN2)
E 24 V supply
F ME bus (CAN1)

ETCHN003: Power supply unit for the supply of the ETC island and ME bus connection

ETCHx004: ETC Motion Control for 4 axes (control)

ETCHIxxx: Input module with 16 or 8 digital inputs

ETCHT000: ME bus terminator module

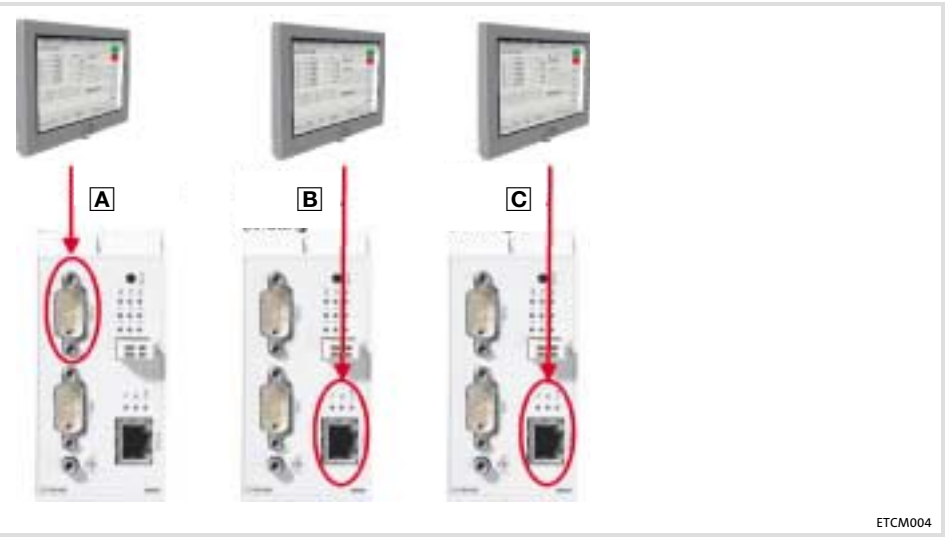
Note!

To terminate the ME bus DIP switch 1 must be set to ON at both the power supply unit ETCHN003 and the bus terminator module ETCHT000.

2.1.3 Connecting ETCHx and PC

Three types of connections are possible between the ETCHx and a PC.

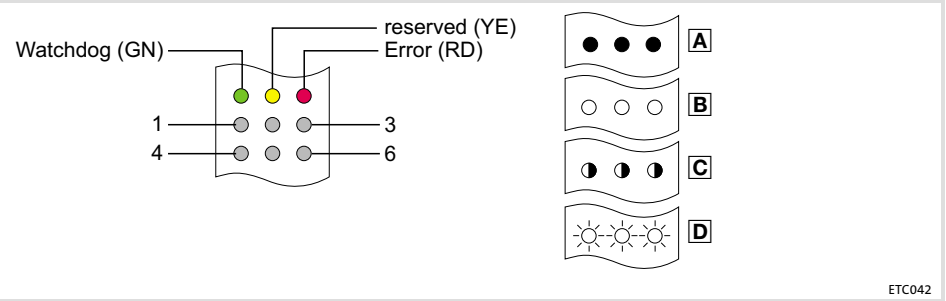
Connection type	Cable version	Description
Serial connection	System cable type EWL 0068 or a comparable RS232 cable with double-sided 9-pin SUB-D socket (for the pin assignment see ETC Hardware Manual)	This connection is only required for commissioning! A free COM port at the PC is connected with the RS232 interface of the ETCHx. The PC communicates via a terminal program with the monitor interface of the ETCHx.
Local connection via Ethernet (pear-to-pear connection)	System cable type EWL 0065, EWL 0066 or EWL 0067 (double-sided RJ45)	A network card in the PC is connected with the Ethernet connection of the ETCHx via a cross-over cable. Communication takes place via the TCP/IP protocol.
Network connection	Ethernet patch cable STP Cat5 (double-sided RJ45)	The ETCHx is connected to a separate or existing network (e.g. Intranet) via an Ethernet connection. Communication takes place via the TCP/IP protocol.



A Connection via serial interface
B Local TCP/IP connection
C Network TCP/IP connection

2.2 Status display

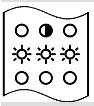
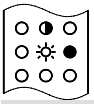
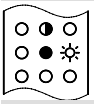
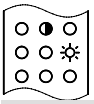
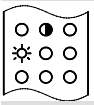
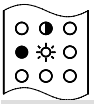
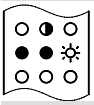
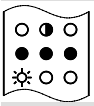
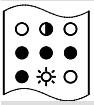
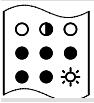
LEDs on the front plate of the ETC report the actual system state. The meanings of the signals differ in the start-up phase and during operation.



A LED on
B LED off
C LED any
D LED flashes

Start-up phase

During start-up a RAM test is carried out. After an error-free RAM test the LEDs 1 ... 6 produce a running indication. Any errors during the boot sequence will be signalled by the following pattern of flashing and indications.

	Checksum error in the internal FLASH-PROM. 3 times fast consecutive flashing. The boot loader is then burned afresh into the internal FLASH-PROM. Occurs always after a boot loader update.
	Error in the last 32 kByte of RAM memory (after an update and writing of the boot loader to the Flash).
	Error in the first 32 kByte ofRAM memory.
	Error when extracting the boot loader.
	Error during the burning of the boot loader into the FLASH-PROM.
	Errors in the RAM memory.
	Error in the last 32 kByte ofRAM memory.
	No firmware loaded or checksum error in the firmware. After 5 times flashing the boot monitor is activated.
	Error when loading the firmware from the FLASH-PROM. After 5 times flashing the boot monitor is activated.
	Error when starting the firmware.

Operation

When the control enters the operating mode after start-up, the following pattern of flashing and indications applies.

LED	Meaning
Watchdog	Watchdog, must always illuminate when running.
Reserved	Without function
ERROR	Flashes after an exception (violation of the control program protection during runtime, exceeding the permitted variable range, division by zero etc.).
LED 1	Flashes at the clock pulse of the coarse interpolator
LED 2	Flashes at the clock pulse of the interpreter
LED 3	Flashes at the clock pulse of the central control
LED 4	Flashes at the clock pulse of the fine interpolator
LED 5	Flashes with each RS232 interrupt or CAN interrupt
LED 6	Flashes at the clock pulse of the PLC cycle time

Note!

The ”flashing” frequency can be so low with short programs, that the LEDs appear dark.

2.3 Commissioning steps (overview)

Stop!

Observe the notes in the chapter ”Initital switch-on” of the ETC Hardware Manual before commissioning the system.

Note!

Only build in and install the PCI control variant ETCPx in step 6.

Step	ETCHC	ETCPC	Description	See
1	X	-	Connect ETCHC via RS232 cable with PC and start ETCHC.	21
2	X	-	Start terminal program on the PC (if required, configure in advance) and activate monitor interface of the ETCHC.	21
3	X	-	If the ETCHC is to be operated in the ”with MMI” operating mode, replace the firmware file on the ETC by the file ”NetBoot.rsc”.	25
4	X	-	Assign the IP address of the ETCHC.	28
5	X	X	If required, install ETC-MMI and ETC-MMI Gateway.	31
6	-	X	Install ETCPC in the PC and install the driver.	31
7	X	X	Start ETC-MMI and ETC-MMI Gateway.	32
8	X	X	Establish connection between ETC-MMI and ETCxC.	34
9	X	X	Parameterise drives via machine constants.	36
10	X	X	Check parameters of the drives.	43
11	X	X	Test drives in inching mode.	44
12	X	X	Create CNC program in the ETC-MMI.	47
13	X	X	Load CNC program into ETCxC, start and test CNC program.	49
14	X	X	If required, install CoDeSys.	51
15	X	X	Start and configure CoDeSys.	51
16	X	X	Create PLC program.	60
17	X	X	Load PLC program into ETCxC, start and test PLC program.	68
18	X	X	Test CNC and PLC program.	72

2.4 Establishing the communication between PC and ETCHx

Note!

The steps described in this chapter only apply to the ETCHx variant (DIN rail variant); they are not required for the ETCPx variant (PCI card).

2.4.1 Starting ETCHx

1. Connect the serial interfaces of PC and ETCHx.
   For this purpose, use the system cable of type EWL 0068 or a comparable RS232 cable with double-sided 9-pin D-Sub socket (for the pin assignment see ETC Hardware Manual).
2. Start the PC.
3. Start the ETCHx by mains connection or initialise the ETCHx by a reset (press the Reset key on the front panel).
   As soon as the ETC firmware has been loaded, the green watchdog LED lights up at the ETCHx .

2.4.2 Starting the terminal program ”HyperTerminal” and activating the monitor interface

For the communication between PC and ETCHx, you require a terminal program. The terminal program ”HyperTerminal” is available in every standard Windows installation.

1. If you have already established a connection with an ETC (i.e. if a connection is already configured), start the connection via **<Start> → Programs → Accessories → Communication → HyperTerminal →** *ConnectionName.ht***.** See next section.
   If no connection has been configured yet, start the HyperTerminal via **<Start> → Programs → Accessories → Communication → HyperTerminal**.
   The program queries different settings of the telephone connection because it is also designed for a modem connection. These settings are not significant in this context.
2. After you have been prompted, enter a name for the connection (for example ”ETC”) and click **OK**.

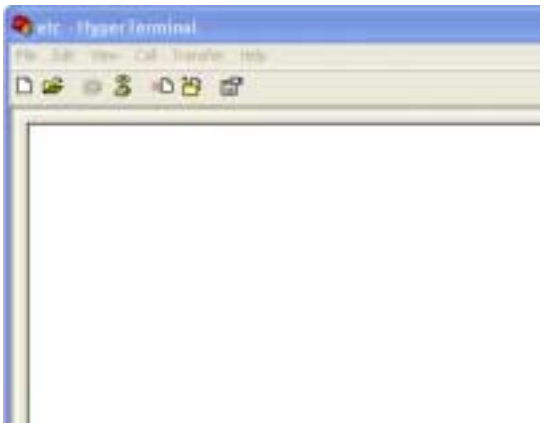
3. In the *”Connect to”* dialogue, select the PC interface via which you want to establish the connection (for example ”COM1”).

ETCM006

4. Click **OK**.
5. In the ”COMx Properties ” dialogue, enter the following data:

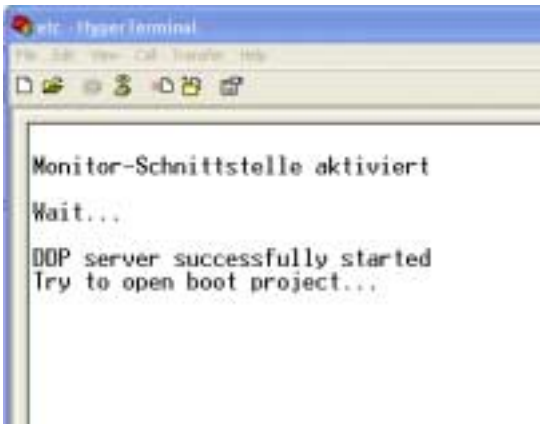
ETCM007

6. Click **OK**.



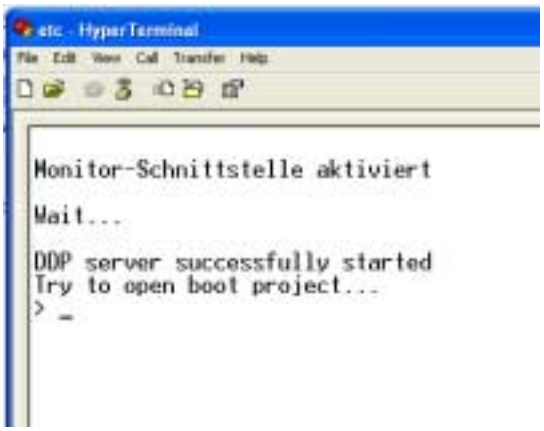
ETCM008

As soon as a connection has been established between the PC and ETC, the LEDs 1 ... 6 at the ETCHx start to flash circulatingly. The message ”Wait Boot Loader” appears and the following window is displayed:



ETCM009

7. Press the **<?>** key until the prompt ”>” appears.



ETCM010

The monitor interface has been started. You can now send commands to the ETC via the keyboard.

Important commands of the monitor program

Command	Meaning
dir [dr:][pattern]	Shows the contents of the specified drive. The flashdisk (sd:) is preset, other possible drives are program storage (ps:), Ram disk (rd:) and floppy disk (fd:), if existing. As pattern, the usual MS-DOS patterns can be used, e.g.: *.din
del [dr:][pattern]	Deletes the specified files on the specified drive. The flash disk (sd:) is preset.
cd [dr:]	Changes to or shows the currently selected drive. The flash disk (sd:) is preset.
format dr:	Formats the current disk and thus deletes all files on this disk. A disk name must be specified.
reboot	Resets the control system. To activate the boot monitor, the ”!” key must be pressed afterwards until the boot monitor responds (this may take 3-4 seconds).
help	Displays the help page with commands of the monitor program.
ver	Displays version string
ipconf	Configures network parameters for Ethernet interface
hwconf	Displays hardware configuration of the control system (CPU type, memory etc.)

Fault elimination

If you cannot establish a connection to the ETCHx by means of the described procedure, proceed as follows:

1. Check the cabling between PC and ETCHx.
2. Check the connection parameters of the terminal program.

A first test of the cable connection can be performed by means of a bridge between the pins 2 and 3 at the socket of the cable (control system side):

► Afterwards, the terminal program is called as usual and any letters can be entered via the keyboard.

► If the entered characters of the keyboard are correctly displayed in the mask of the terminal program (the characters are sent back by the bridge as echo), the cable must be checked for exchanged pins (pin2/3), for short circuit and the connection must be checked for correct parameter setting.

2.4.3 Setting the operating mode of the ETCHx

The ETCHx can be operated in two operating modes:

- Variant "Standalone" (delivery variant)
  - Directly after the voltage has been applied, the control system loads the firmware.
  - The control system executes a fixed program.
- Variant "with MMI"
  - After the voltage has been applied, the control system waits until the firmware is loaded by a PC via any MMI program (e.g. ETC-MMI).
  - Variable user programs can be executed.

Comparison of the operating modes

	Standalone (delivery)	With MMI
Use	The control system operates a machine without further coupling to a PC.	The control system operates a machine with coupling to a user interface, which is used for loading programs and operating the control system.
User program	A fixed program runs in the control system.	Variable user programs can be loaded into the control system.
Installed firmware on the control system	Boot loader firmware	Boot loader loader
Installation of the firmware	The firmware is loaded and installed in the EEPROM by the boot loader.	The Loader is loaded and installed in the EEPROM by the boot loader.
Behaviour of the control system after the voltage has been applied	After the voltage has been applied, the control system automatically starts with its firmware.	After the voltage has been applied, the control system waits until the firmware is loaded via the PC.
LED display after the voltage has been applied	After the firmware has been started successfully, the green watchdog LED lights up.	The successful start of the loader is indicated by circulating LEDs (1 ... 6).
Starting the firmware	Before the start, the firmware is loaded from the EEPROM of the control system into the RAM and started.	The firmware is located on the PC and is loaded into the RAM of the control system and started when the MMI user interface is started. After the firmware has been started successfully, the green watchdog LED lights up.

Set the "Standalone" operating mode

The "Standalone" operating mode is factory-set.

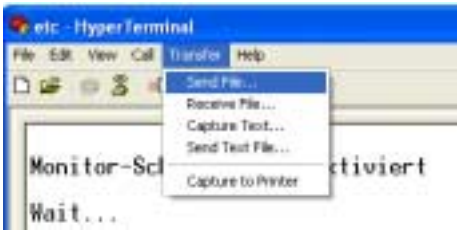
Note!

If the ETCHx has already been operated in the "with MMI" operating mode and is to be reset to the "Standalone" operating mode, the file "NetBoot.rsc" (loader) must be deleted in the control system and the firmware file ETCHC_A.rsc (ETCHC) or ETCPC.rsc (ETCPC) is to be transferred instead. For this purpose, proceed as described in the following section; the files that are to be exchanged are the only difference.

Set the "With MMI" operating mode

For the "With MMI" operating mode, the firmware file must be replaced by the file "NetBoot.rsc" (Loader) on the ETCHx.

1. To delete the firmware, enter the following in the HyperTerminal: **del sd: etc*.rsc**. Afterwards, press the **<Enter>** key.
2. To transfer the Loader to the ETC, activate the menu item **Transfer → Send File** in the HyperTerminal.



ETCM021

3. Specify the path to the file "NetBoot.rsc" and select the protocol "Zmodem with Crach Recovery".

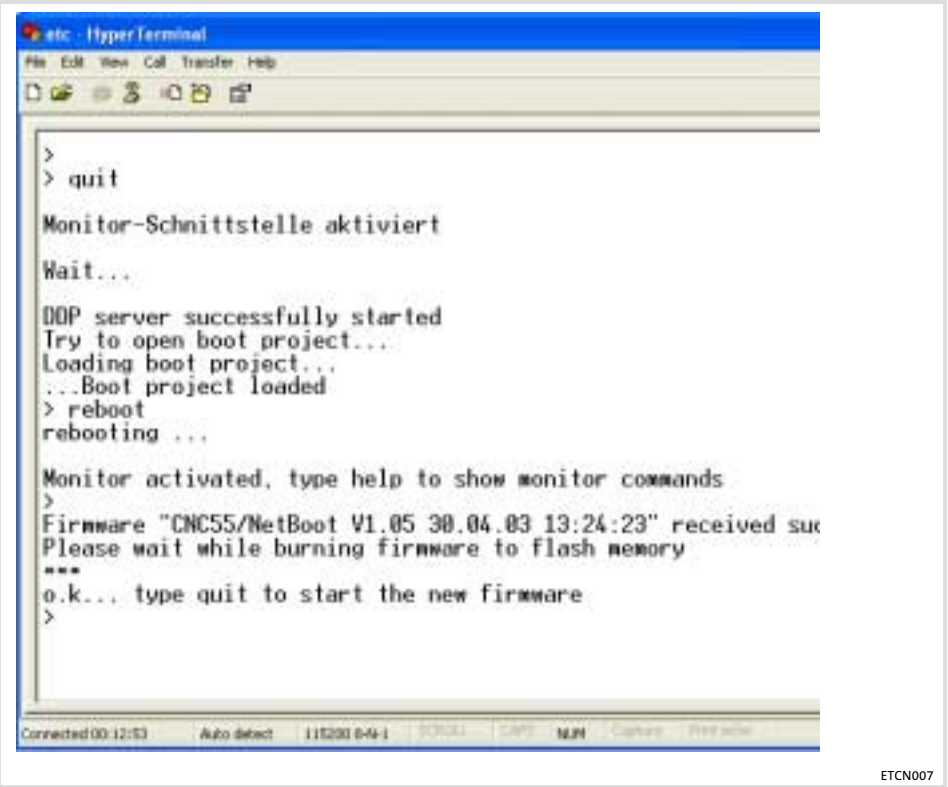
ETCM022

4. Click on **Send**.

   While the file is being loaded into the control system, the following figure is displayed. On the ETC front plate, the LEDs 1 ... 6 flash circulatingly.

ETCM023

After loading is complete, the following figure is displayed:



The control system waits until the firmware is loaded from the PC; on the ETC front plate, the LEDs 1 ... 6 flash circulatingly.

5. In the HyperTerminal window, enter **quit** and confirm the command with **<Enter>** .

   The firmware starts. On the ETC front plate, the green watchdog LED lights up.

2.4.4 Assigning the IP address of the ETCHx

For communication via a network or a local Ethernet cable, the ETCHx requires a unambiguous IP address (with subnet mask) that matches the other nodes. When the control system is delivered, it has a specific, but random IP address.

The IP address of the ETCHx is assigned via the monitor interface (as described in the following). This IP address is requested when connections are established in the ETC-MMI Gateway (📖 34).

Specify the IP address according to the following general rules for IP addresses:

Separate (local) network segment

If the control system and PC are connected with a local network segment or via a direct Ethernet connection (without Internet/Intranet connection), assign IP addresses of class B or class C to the two network stations.

- ► Addresses of class B are in the range from 128.0.0.0 to 191.255.255.255. The corresponding subnet mask is 255.255.0.0 (default).
- ► Addresses of class C are in the range from 192.0.0.0 to 223.255.255.255. The corresponding subnet mask is 255.255.255.0 (default).

The gateway address is not relevant here. You can enter a free IP address of the subnetwork (not 0.0.0.0).

Example	IP address	Subnet mask	Gateway address
This address must be assigned to the ETCHC and specified as connection parameter in the ETC-MMI Gateway 📖 34:	181.16.1.10	255.255.0.0	181.16.1.1
This address must be specified in the network connection of the PC (TCP/IP settings):	181.16.1.11	255.255.0.0	181.16.1.1

Note!

Do not use any leading zeros in the IP addresses. Otherwise, the IP address will be interpreted as octa decimal number.

Existing network segment

If ETCHx and PC are to be connected with an existing network segment, the addresses and the subnet mask must be assigned by the network administrator.

Note!

If the Ethernet adapter supports 100Base-T technology, ”Auto Select” or ”10Base-T” must be set for the ”Media Type” property.

MAC address

Like any other device with Ethernet controller, the ETCHx receives an unchangeable and worldwide unique physical Ethernet address, also called MAC ID (Media Access Control Identity), from the manufacturer. It can be used for addressing on the hardware level. The address has a fixed length of 6 bytes (48 bits) and contains an address type, an identification of the manufacturer and a serial number.

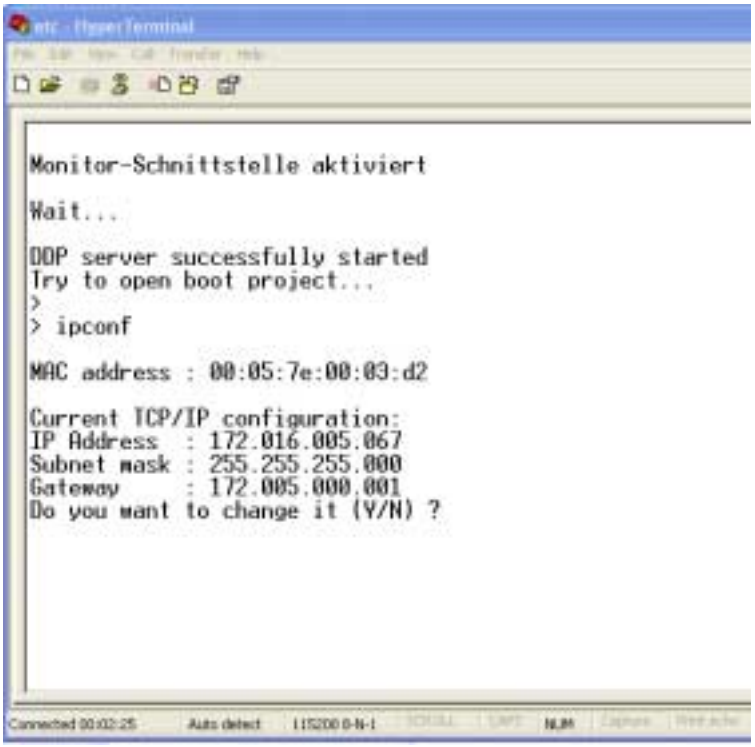
MAC-ID of the ETC: 00:05:7e:xx:xx:xx (with xx:xx:xx = serial number).

Assigning an IP address

The IP address of the ETCHx is specified via the monitor interface.

1. In the HyperTerminal, enter the command **ipconf** and confirm the command with **<Enter>**.

   The MAC-ID and the current address setting of the ETCHx are displayed.



ETCM011

2. Press the **<Y>** key and change the settings. Follow the instructions of the monitor interface.

   Select an IP address according to the rules specified at the beginning of the section.

   The HyperTerminal program can now be quit.

Note!

Note down the IP address because it must be specified when creating connections in the ETC-MMI Gateway (📖 34).

2.5 Configuring ETC-MMI and ETC-MMI gateway

The program ”ETC-MMI” is used for the following tasks:

- ► Configuring the control system
- ► Operating and monitoring the control system
- ► Maintenance of the control system and error diagnosis

The MMI gateway is the communications program between Windows applications and ETC control systems. Different applications such as MMIs, configuration tools or OPC servers can establish connections to one or more control systems at the same time.

The gateway implements all required mechanisms for access control, error handling and diagnostics and supports control-specific hardware drivers and communication protocols. By means of the configuration tools, new connections can be added and existing connections can be edited.

The ETC-MMI Gateway is installed during the installation of the Lenze ETC-MMIs. It can also be installed as a separate application. In this case, it is available as communication channel for other applications.

For further information, please refer to the chapters ”ETC-MMI Gateway” (🕮 276) and ”ETC-MMI” (🕮 286).

2.6 Installing ETC-MMI

Stop!

Only install the PCI control variant ETCPx after installing the ETC-MMI and before starting the ETC-MMIs.

Note!

The ETC-MMI Gateway is installed during the installation of the Lenze ETC-MMIs

1. In Windows File Explorer, open the program ”setup.exe” on the ETC-MMI installation CD.
2. Follow the instructions of the installation program. The following will be requested:
   – Demo or standard installation
   – Control type (ETCHC or ETCPC)
   – IP address of the ETCHC (📖 28) or index of the ETCPC (indexes of the PCI cards are in the range 0 ... 9 and are automatically assigned by the driver. The first ETCPx has the index 0).

   In the start bar, the entry ”Lenze ETC-MMI” is created. It can be used to start the application.
3. For an ETCHC: Check that the TCP/IP protocols are installed in the network properties (**Start → Settings → Control Panel → Network**).

   For an ETCPC: Install the ETCPC plug-in card in the PC. Observe the notes in the ETC Hardware Manual.

2.6.1 Building in and installing the ETCPx

1. If you use an ETCPx plug-in card as control system, install it in the PC now. Please observe the notes in the ETC Hardware Manual.
2. Start the PC and open the BIOS settings.

   For Phoenix BIOS: set Advanced\Plug & Play O/S to [NO].

   For other BIOS, check the corresponding entry, e.g. PNP OS on [NO].

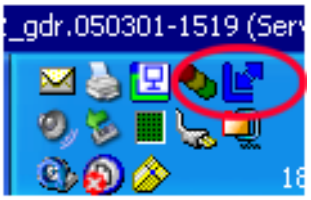
   After the start of the operating system, the hardware wizard detects the new hardware and queries the driver.
3. Install the ETCPx driver from the ETC-MMI CD (...\wdm2000\ETCPC.inf)
4. Restart the PC.

2.7 Starting ETC-MMI

1. Start the ETC-MMI via **<Start> → Programs → Lenze → ETC**

ETCN011

The ETC-MMI Gateway is automatically started. The application can be seen on the task bar:



ETCN001

Note!

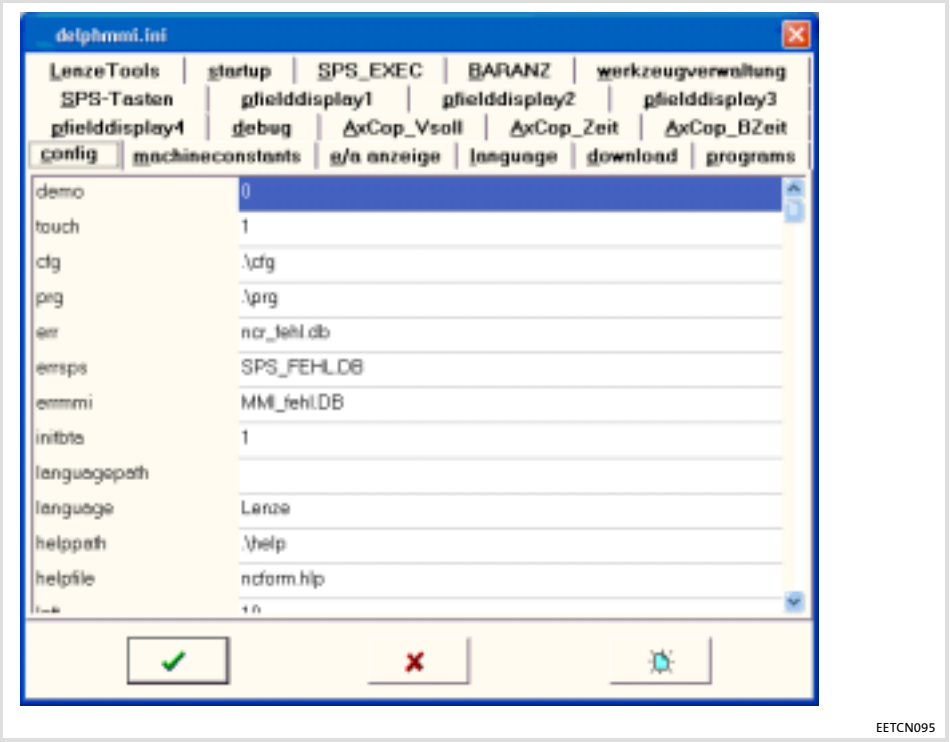
It can be defined which operating mode is displayed when the user interface is started. (📖 327).

For a detailed description of the MMIs, refer to chapter ”ETC-MMI” (📖 286).

For a detailed description of the MMI gateways, refer to chapter ”ETC-MMI Gateway” (📖 276).

2.7.1 Switching the language in the ETC-MMI

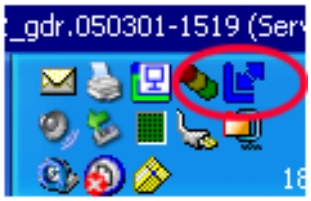
1. In the ETC MMI window, press **<F12>** (diagnostics).
   The MMI window opens in the ”Diagnostics” operating mode.
2. In the ETC-MMI diagnostics window, press **<F8>** (MMI-config.).
   The window ”delphmmi.ini” opens (📖 283).



3. Specify the language in the line *Language*:
   German: ”Lenze”
   English: ”Lenze_gb”
4. Close the window with **<Enter>**.
5. Quit and restart the ETC-MMI.

2.7.2 Establishing a connection between ETC-MMI and ETC

1. On the task bar, click on the ETC-MMI Gateway icon.



ETCN001

A menu opens.

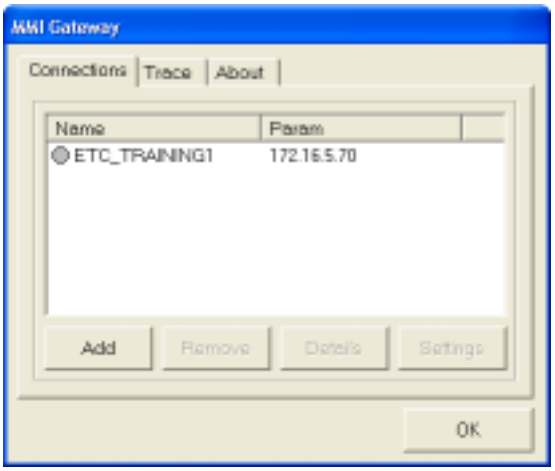
ETCN002

Settings: Start configuration interface.

About: Display version and manufacturer information.

Exit: Close gateway (if there are active connections to an application, a warning is displayed).

2. Click on **Settings**.



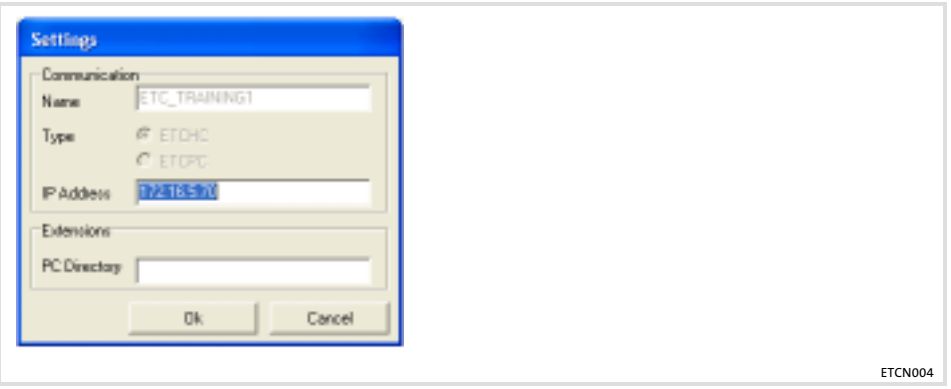
ETCN003

Connections: List of the already configured connections. A green LED next to the connection name indicates an active connection, via which the messages and/or cyclic data are transferred. Via the **Settings** button, you can edit the communication parameters of the selected connection (📖 278).

Trace: In the case of faults in the communication, you can activate trace logs on this tab (📖 281).

About: Display version and manufacturer information.

3. To create a new connection, click on **Add** on the ”Connection” tab.



4. Specify a name for the connection.

   To enable an application to communicate with a control system via the ETC-MMI Gateway, each connection must be assigned an unambiguous name. You can choose any name. Assign e.g. consistent names ”ETC0”, ”ETC1” ... or application-specific names ”ramp”, ”laser control” etc. A name can contain up to 31 ASCII characters (”A ... Z”, ”a ... z” , ”0 ... 9”).

5. Select a connection type:

► UDP connection for ETCHx control systems

  Ethernet communication via DPR/UDP/IP protocol.

► PCI connection for ETCPx control systems

  DPR communication via a WDM driver (ETCPC.sys, in the scope of supply of the ETCPC).

6. Enter a communication parameter in the ”IP address” field:

► ETCHC = IP address (📖 28)

► ETCPC = Index of the PCI cards (indexes of the PCI cards are in the range 0 ... 9 and are automatically assigned by the driver. The first ETCPx has the index 0).

7. Close the dialogue via **OK**.

   Your entries are saved in the file ”mmigtway.ini” (📖 283). If a connection with the same parameters already exists, the error ”Gateway cannot change the parameter” is reported.

2.8 Parameterising drives via machine constants

The properties of the drives must be parameterised both in the drive itself and in the control system.

In the control system, the properties are assigned via machine constants (MCs). A machine constant consists of a keyword and the corresponding values; for example ”MK_VMAX 20”.

The machine constants are saved on the PC in a text file with the extension .mk (e.g. ETCHC.mk for the ETCHC) and are loaded into the control system via the ETC-MMI. The machine constants can be changed after loading via a dialogue in the ETC-MMI.

2.8.1 Overview of the most important machine constants

The machine constants listed in the following are mandatory for the operation of the drive. In relation to the drives, they are only a subset of the required parameterisation. For a detailed description of all machine constants, refer to chapter ”Machine constants” (📖 190).

Stop!

The control system only initialises the drives via its machine constants and determines path-related parameters. The control-related parameters, the safety functions of the drives and the parameters of the reference run must be parameterised in the drive itself by means of the GDC (Global Drive Control).

Test setting

MC keyword	No. of values	Values	Meaning
MK_TEST_OHNEMECHANIK	1	0, 1	**0:** ”Achsrechner” (”Axis computer”) of the NC firmware works and considers the actual position counter of the drive. **1:** ”Achsrechner” (”Axis computer”) of the NC firmware works, but simulates the actual position counter of the drive (actual position = desired position for the test of the NC function without mechanics).
MK_SPS_DUMMY	1	0, 1, -1	**0:** with integrated PLC function **1:** without integrated PLC function **-1:** without integrated PLC function, releases are reset

Hardware configuration

MC keyword	No. of values	Values	Meaning
MK_CANDRIVES	12	-1, 0 ... 11	Assignment of the **axis number** 0 ... 11 to the **CAN node address** in the order of the CAN node address 1 ... 12 **-1:** No axis number is assigned to the node address **0 ... 11:** An axis number is assigned to the node address
MK_APPLACHSIDX	18	-1, 0 ... 11	Assignment of the **axis number** 0 ... 11 to the **axis description** in the order of the axis description X ,Y, Z, C, U, V, W, A, B, u, v, w, x, y, z, a, b, c **-1:** No axis number is assigned to the axis description **0 ... 11:** An axis number is assigned to the axis description
MK_ACHSENART	12		Assignment of the **axis type** to the **axis number** in the order of the axis number 0 ... 11
			Bit 0 Axis type
		xxxxxxx0	Linear axis
		xxxxxxx1	Rotation axis
			Bit 1 Limit switch
		xxxxxx0x	Observe HW limit switch
		xxxxxx1x	Ignore HW limit switch
			Bit 3/2 Axis type
		xxxx00xx	Normal axis
		xxxx01xx	Spindle
		xxxx10xx	Measurement axis
		xxxx11xx	Spindle and measurement axis
			Bit 5/4 Only for xxxxxxx1, rotation axis
		xx00xxxx	Rotation axis with absolute positioning
		xx01xxxx	Modulo 360° axis - sign indicates the direction
		xx11xxxx	Modulo 360° axis - shortest distance is travelled (ETCHC only)
			Bit 6 Gantry axis
		x0xxxxxx	Normal axis
		x1xxxxxx	Gantry axis (synchronous axis with mechanical coupling)
			Bit 7 Handwheel
		0xxxxxxx	Normal axis
		1xxxxxxx	Handwheel

Software configuration

MC keyword	No. of values	Values	Meaning
MK_CANOPEN_BAUDRATE	2	0, 10... 1000	1st parameter for CAN1 2nd parameter for CAN1 **0:** no device connected **>0:** baud rate for CAN-Open in kB
MK_DELTAT	1		Rough interpolation cycle in [ms]

Setting of the axes

MC keyword	No. of values	Values	Meaning
MK_IMPULSE	12		Number of impulses per [MK_WEG] (after the quadruplication!) in the order of the axis number 0 ... 11
MK_WEG	12		Distance in [mm] or [degree] which corresponds to the value of [MK_IMPULSE] in the axis computer in the order of the axis number 0 ... 11

Axis-related limit values

MC keyword	No. of values	Values	Meaning
MK_MODVMAX	12		Modal axis speed (manual traversing) [m/min] or [U/min] in the order of the axis number 0 ... 11
MK_VMAX	12		Max. axis speed [m/min] or [U/min] in the order of the axis number 0 ... 11
MK_BESCHL	12		Acceleration ramp [m/s^2] or [U/s^2] in order of the axis number 0 ... 11
MK_BREMS	12		Deceleration ramp [m/sec^2] or [U/s^2] in the order of the axis number 0 ... 11
MK_T_BESCHL	12		Damping time constant for acceleration and deceleration ramps [ms] in the order of the axis number 0 ... 11

Path-related limit values

MC keyword	No. of values	Values	Meaning
MK_VBAHNMAX	1		Max. material speed [m/min]
MK_BAHNBESCHL	1		Acceleration ramp [m/s^2]
MK_BAHNBREMS	1		Deceleration ramp [m/s^2]
MK_T_BAHNBESCHL	1		Dampening time constant for deceleration and acceleration ramps [ms]

2.8.2 Machine constant file ETCxC.mk

In the control variant "With MMI", the file ETCxC.mk is loaded into the control system ETCxC when the ETC-MMI is started. In the "Standalone" variant, it is detected that the machine constants have already been loaded.

The following example of machine constants is an excerpt from the file ETCxC.mk with preset machine constants, which must be adapted for the specific application.

Test setting

MC keyword	No. of values	Values
MK_TEST_OHNEMECHANIK	1	0
MK_SPS_DUMMY	1	0

Hardware configuration

MC keyword	No. of values	Values
MK_CANDRIVES	12	0, 1, 2, 3, -1, -1, -1, -1, -1, -1, -1, -1
MK_APPLACHSIDX	18	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17
MK_ACHSENART	12	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Software configuration

MC keyword	No. of values	Values
MK_CANOPEN_BAUDRATE	2	500, 1000
MK_DELTAT	1	2

Setting of the axes

MC keyword	No. of values	Values
MK_IMPULSE	12	65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536
MK_WEG	12	10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10

Axis-related limit values

MC keyword	No. of values	Values
MK_MODVMAX	12	10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10
MK_VMAX	12	20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20
MK_BESCHL	12	2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
MK_BREMS	12	2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2
MK_T_BESCHL	12	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Path-related limit values

MC keyword	No. of values	Values
MK_VBAHNMAX	1	20
MK_BAHNBESCHL	1	2
MK_BAHNBREMS	1	2
MK_T_BAHNBESCHL	1	0

2.8.3 Notes on loading the MK file into the control system

Make sure that the number of parameters in the file of machine constants corresponds to the number of axes (12) in the operating system of the control system.

When the machine constant file is loaded into the control system, 3 cases are possible:

- The number of parameters in the file **equals** the maximum number in the operating system.

  The parameters overwrite the saved machine constants in the control system.

- The number of parameters in the file **is greater** than the maximum number in the operating system.

  The parameters overwrite the saved machine constants in the control system up to the maximum number. An error message is generated for each parameter of the file that is not adopted.

- The number of parameters in the file **is smaller** than the maximum number in the operating system.

  The parameters overwrite the saved machine constants in the control systems. The missing machine constants up to the maximum number are completed by the entries that already exist in the control system.

Note!

If the control system is in an error status (display in the ETC-MMI in the SETUP operating mode: ”NCR: ERROR”), the loaded machine constants are not adopted in the control system.

The machine constants have only been adopted correctly when the control state ”Idle” (display in the ETC-MMI in the SETUP operating mode: ”NCR: STEHT” (”NCR: STANDS”)) is reached.

2.8.4 Example for adapting a machine constant file

The drive configuration has 3 drives with the following properties:

	Drive 1	Drive 2	Drive 3
Axis number	0	1	1
CAN node address	8	7	3
CAN baud rate	500 kB		
Axis description	c	X	(X')
Axis type	Rotation axis	Linear axis with handwheel	Gantry axis for drive 2 with the same properties
Resolution pulses	65536	65536	like drive 2

The machine constant file must look as follows:

Test setting

MC keyword	No. of values	Values
MK_TEST_OHNEMECHANIK	1	0
MK_SPS_DUMMY	1	0

Hardware configuration

MC keyword	No.of values	Values
MK_CANDRIVES	12	-1, -1, **1**, -1, -1, -1, **1**, **0**, -1, -1, -1, -1
MK_APPLACHSIDX	18	**1**, -1, -1, **0**, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1
MK_ACHSENART	12	**1**, **192**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Software configuration

MC keyword	No. of values	Values
MK_CANOPEN_BAUDRATE	2	500, 1000

Setting of the axes

MC keyword	No. of values	Values
MK_IMPULSE	12	65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536, 65536

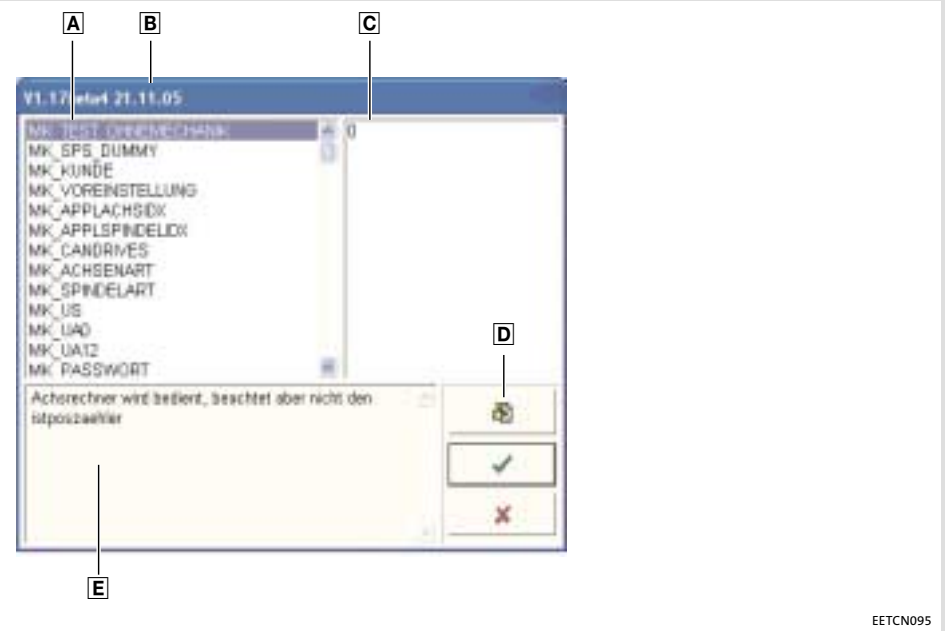
Explanation

To define a synchronous axes, the same axis number is entered at 2 CAN note addresses in MK_CANDRIVES. This creates a forced coupling of the axes of the two CAN node addresses. The CAN axis with the lower node address automatically is the master axis, the axis with the higher node address and the same axis number is the slave axis (synchronous axis).

The gantry axis (X', in MK_ACHSENART) is a special case of the synchronous axes. It is a mechanical forced coupling of 2 axes. Thus, the gantry axis is not entered as an axis of its own in MK_CANDRIVES, but as a reference to the X axis.

2.8.5 Adapting machine constants in the ETC-MMI

1. In the ETC MMI window, press **<F12>** (diagnostics).
   The MMI window opens in the ”Diagnostics” operating mode.
2. Press **<F6>** (machine const.).
3. Press **F6** (Change current MCs) again.
   The current machine constants are loaded.



A: Available machine constants
B: Date and version of the NC firmware
C: Value of the selected machine constant
D: Accept changed value
E: Short description of the selected machine constant

4. In the field of the available machine constants A, select the machine constant that you want to adapt.
5. To change the value of the selected machine constant, press the **<Space bar>** and enter the new value in the field C.
6. To accept the value, press the key D.
7. After you have made all changes, press **<Enter>**.
   The changes are transferred to the ETC and updated in the MK file.

Test setting

Note!

Basically, the operation of the CNC program is also possible:

► without connected mechanics and drives.

This is achieved by setting the machine constant MK_TEST_OHNEMECHANIK=1.

► without a PLC program.

This is achieved by setting the machine constant MK_SPS_DUMMY=1.

For the following tests with mechanics, the above-mentioned machine constants must be set as follows:

MC keyword	No. of values	Values
MK_TEST_OHNEMECHANIK	1	0
MK_SPS_DUMMY	1	1

2.8.6 Checking the parameters of the drives

After the machine constants have been adapted, the connected drives can be checked via their tolerance margin in the diagnostics.

1. In the ETC MMI window, press **<F12>** (diagnostics).
   The MMI window opens in the ”Diagnostics” operating mode.

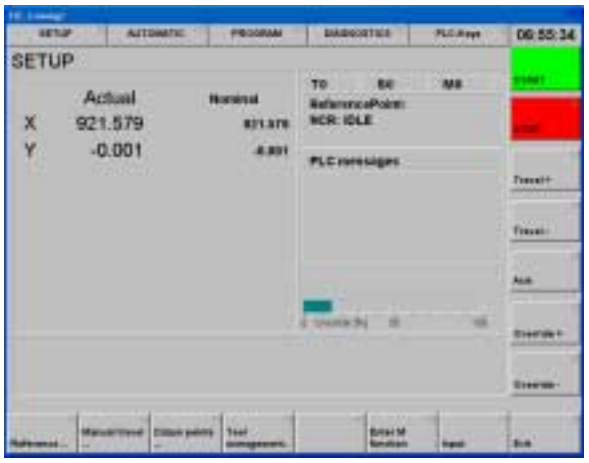
ETCN012

Correctly parameterised drives show an alternating tolerance margin around the zero point.

2.8.7 Testing the drives in inching mode

After the machine constants have been adapted, the drives must be tested in inching mode. Check whether the configured drives behave according to the specifications.

1. In the ETC MMI window, press **<F9>** (setup).
   The MMI window opens in the ”Setup” operating mode.
2. Press **<F2>** (Manual travel).
3. In the submenu, press **<F1>** (Modal travel).
   An individual drive is selected. The selected drive is indicated by the setpoint with a green background.



ETCN011

4. In the submenu, alternately press **<S3>** (Travel +) and **<S4>**(Travel -).
   In inching mode, the selected drive rotates in positive or negative direction.
5. To change the speed of the drive, press **<S6>** (Override +) or **<S7>** (Override -).
6. To select the next axis for traversing, press **<S5>** (axis).
   The green setpoint display switches.

2.9 CNC programming according to DIN 66025

The following description of functions according to DIN 66025 is an excerpt from the chapter ”CNC programming” (📖 87).

2.9.1 G-functions

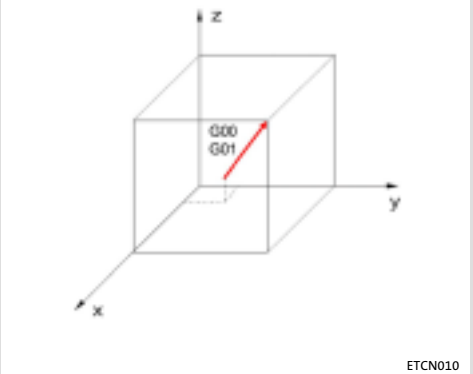
G-functions define geometric preparatory functions for the operation of the axes. On principle, a DIN block with a G-function has the following structure:

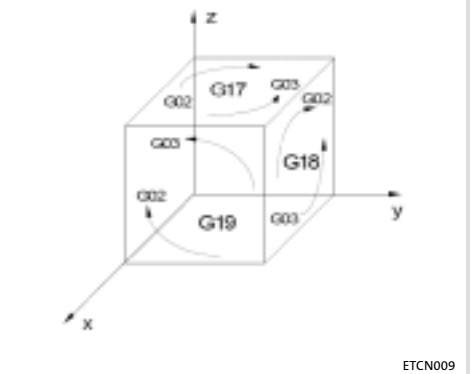
The letter ”G” follows the number of the G-function. This is followed by the parameters, which consist of a letter and the corresponding value.

G (number) [parameter (value)] [parameter (value)] ...

The following shows some examples of G-functions:

g	Meaning	Parameter	Example with parameter and values
00	Linear interpolation, high rate	AXES R D F E L	G1 X0 Z10 C0 F2
01	Linear interpolation		
02	Circular interpolation, clockwise	AXES D I J K R F E L	G3 X40 Y40 R100 D-60 K2
03	Circular interpolation, counterclockwise		
17	Plane selection XY for circular interpolation	--	G17
18	Plane selection XZ for circular interpolation	--	G18
19	Plane selection YZ for circular interpolation	--	G19





Parameter for G00, G01

Parameter	Meaning
AXES	Target point coordinates of the linear axes X, Y, Z, A, B, C, U, V, W, x, y, z, a, b, c, u, v, w Note: In a G-function, only the axes X, Y, Z, A, B, C, U, V, W or axes x, y, z, a, b, c, u, v, w may be used.
d	Max. path deviation in the target point for grinding corners with the following linear interpolation.
R	Radius, with which the following linear interpolation (G0, G1) should be connected.
f	High rate speed (G00) or feed speed (G01) on the path (G31) or each axis (G30), also for all following linear interpolations.
E, L	Selection of feed speed via speed (E) and increment (L). F=E*L.

Parameter for G02, G03

Parameter	Meaning
AXES	Target point coordinates of the three linear principal axes (default X,Y,Z), as well as start or target point coordinates of the three rotative secondary axes (default A,B,C).
d	Radius change on reaching the target point
I	Center coordinates of the first principal axis (X) or number of additional full circles
J	Center coordinates of the second principal axis (Y) or number of additional full circles
K	Center coordinates of the third principal axis (Z) or number of additional full circles
R	Interpolation radius
f	Path speed
E, L	Selection of feed speed via speed (E) and increment (L). F=E*L.

2.9.2 M-functions

M-functions define machine commands. They are edited in the CNC and also sent to the PLC. In the PLC, they are evaluated and trigger the programmed function in cooperation with the CNC.

M-functions do not have any parameters. The coordination between the PLC and the CNC takes place via the interface signals PLC-NC (📖 240).

The following shows some examples of fixed M-functions. All other M-functions are available to the programmer.

Parameter	Meaning
14	Start of the profile, switches the summation of the path length in the parameter field P551 on. When M14 is used in an online program (program with reloading of program parts), the design of the program is blocked until the end of the profile (M15/M16/M21) is in the memory.
15	End of profile, switches the summation of the path length in P551 off (begin of idle travel).
30	Program end

2.10 Creating a CNC sample program

2.10.1 Calling the text editor in the ETC-MMI

1. If required, start the ETC-MMI via **<Start> → Programs → Lenze → ETC**.
   The ETC-MMI window opens.

ETCN011

2. In the ETC-MMI window, press **<F11>** (program).
   The MMI window opens in the ”Programming” operating mode. It shows the text editor for entering the CNC program.
3. Press **<F8>** (Graphics) and afterwards **<F7>**(Graphics on/off).
   The window is split. The text editor is displayed on the left, the display field for the graph (graphic area) on the right. In the top left of the text editor, the cursor flashes.

ETCN014

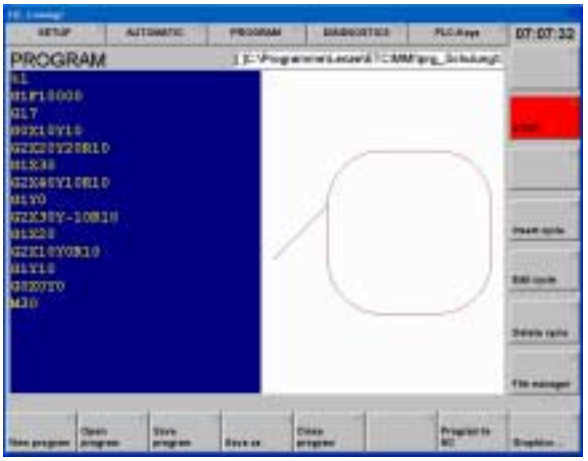
2.10.2 Entering and saving a CNC program

We create a program for a profile with rounded reference points.

Starting point: The cursor flashes in the top right of the text editor and the graphic area is displayed.

1. Enter the CNC program in the text editor according to the following table:

Line	Input in the text editor	Meaning	Display in the graphic area
1	%1	Determine program number	
2	G1 F10000	Determine travelling speed	
3	G17	Determine plane	
4	G0 X10 Y10	Travel to starting point at high rate	creates a straight line from the zero point to the starting point
5	G2 X20 Y20 R10	Execute a section of the square profile	creates a quarter circle
6	G1 X30		adds a straight line
7	G2 X40 Y10 R10		adds a quarter circle
8	G1 Y0		adds a straight line
9	G2 X30 Y-10 R10		adds a quarter circle
10	G1 X20		adds a straight line
11	G2 X10 Y0 R10		adds a quarter circle
12	G1 Y10		adds a straight line to a closed square profile
13	G0 X0 Y0	Travel to zero point at high rate	
14	M30	End of program	



ETCN015

2. Press **<F4>** (Save as).
3. As program name, enter e.g. **test.din**.
4. Press **<Enter>**.

2.10.3 Loading the CNC program into the control system and starting it

Load program into the control system

Starting point: PLC program is displayed in the editor.

1. Press **<F7>** (Program to NC).
2. Select the program; e.g. **test.din**.
3. Press **<Enter>**.
   The program is transferred to the control system ETCxC.

Start program in the control system

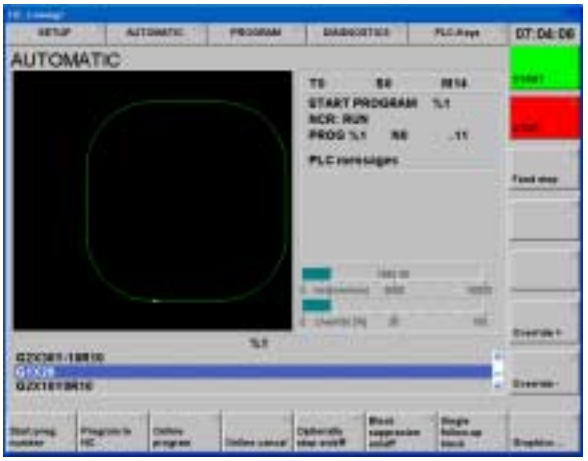
Starting point: The cursor flashes in the top right of the text editor and the graphic area is displayed. In the control system, the program (e.g. **test.din**.) is loaded.

Note!

On principle, operation of the CNC program is possible without connected mechanics and drives and also without a PLC program. For this purpose, the following machine constants must be set:

- ► MK_TEST_OHNEMECHANIK=1
- ► MK_SPS_DUMMY=1

1. Press **<S1>** (Start).
   The drives start and the profiles of the rectangle are executed (if the graphic is not switched on, the numbers of the set and actual positions change). The individual commands are shown in the lower part of the window.



ETCN016

The drawn straight line from the zero point (0/0) to the starting point (10/10) is now executed when the drive is also in the zero point. Otherwise, the control system traverses the drives on a straight line from the current point to the starting point.

2. To change the material speed, press **<S6>**/**<S7>** (Override +/-).
3. To stop the drives, press **<S2>** (Stop).

2.10.4 Extending the CNC program

The sample program (”test.din”) executes a circular profile. The actual target of a CNC program is to switch on a tool while executing a program. Thus, the sample program is extended by the corresponding M-functions M14 and M15, which lift or lower the tool when the profile is executed. The M-functions are programmed in a separate PLC program.

M-function	Target function	Programming of a separate PLC program
M14	Lower tool	An output for lowering the tool is to be set. Switchover to the next command only after one of the lower limit switches has been activated.
M15	Lift tool	An output for lifting the tool is to be set. Switchover to the next command only after one of the upper limit switches has been activated.

Starting point: The cursor flashes in the top right of the text editor and the graphic area is displayed. In the control system, the program (e.g. **test.din**.) is loaded.

1. Press **<F2>** (Open program).
2. Select the program (e.g. **test.din**) and press **<Enter>**
3. Press **<F7>** (Accept).
4. Add the M-functions according to the following table.

Input in the text editor	Meaning
%1	
G1 F10000	
G17	
M15	Lift tool
G0 X10 Y10	
M14	Lower tool
G2 X20 Y20 R10	
G1 X30	
G2 X40 Y10 R10	
G1 Y0	
G2 X30 Y-10 R10	
G1 X20	
G2 X10 Y0 R10	
G1 Y10	
M15	Lift tool
G0 X0 Y0	
M30	

5. Press **<F3>** (Save program).

Exit ETC-MMI

1. Press **<F9>** (Setup).
2. Press **<F8>** (Exit program).

2.11 ETC PLC programming with CoDeSys

The tool ETC-CoDeSys is a complete integrated development environment for creating and testing PLC programs for the ETC.

It is based on the commonly used program package CoDeSys with the special extensions for ETC control systems.

Note!

For further information, refer to the chapter ”PLC programming” (📖 336).

2.11.1 Installing CoDeSys

1. Place the CoDeSys setup CD into your CD-ROM drive.
   If the autostart feature for the CD-ROM drive has been enabled in Windows, setup will start automatically.
2. If setup does not start automatically:
   – Select the menu item ”Run” from the start menu.
   – Enter the drive letter of your CD-ROM drive followed by ”setup.exe” (e.g. ”d:\setup.exe”) into the command line and confirm with ”OK”.
3. When selecting the target directory please create a new directory ”CoDeSys” under ”c:\program files\Lenze\ETC”.
4. Follow the instructions of the setup program.
5. Change the installation path to ”c:\program files\lenze\etc\CoDeSys2”.

Stop!

The file ”codesys.ini” in the installation directory may be write-protected after installation from CD. The write-protection must be cancelled, otherwise the CoDeSys settings will not be saved.

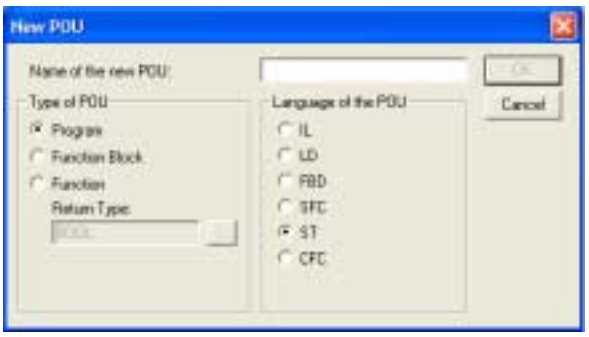
2.11.2 Configuring the control system in the ETC-CoDeSys

This chapter describes the general procedure for control configuration as a precondition for creating a program with the ETC-CoDeSys. An ETCHC is used as an example.

This chapter can be skipped if the sample program (📖 60) is called because it already contains the described settings.

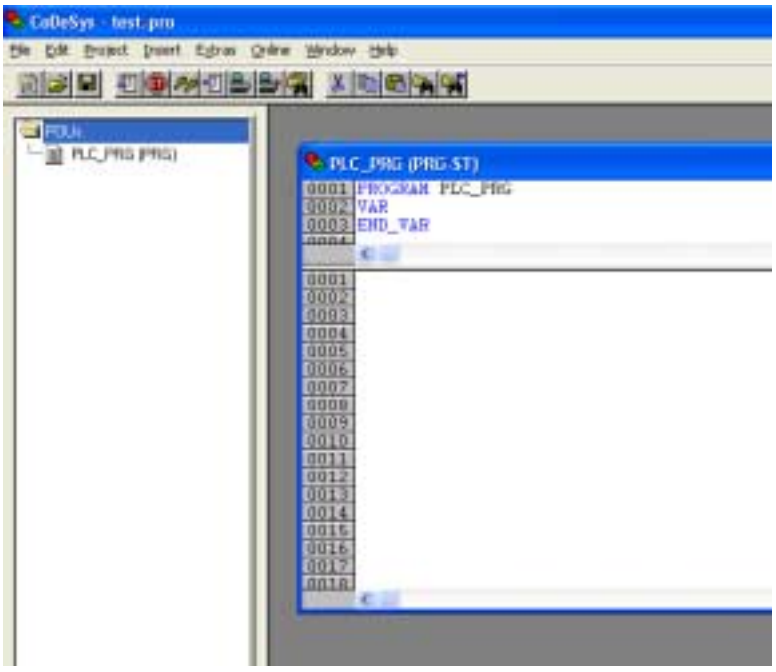
Start ETC-CoDeSys and create new project

1. Start the ETC-CoDeSys via **<Start> → Programs → Lenze → CoDeSys**.
2. If required, you can change the language in the CoDeSys:
   – **Projekt → Optionen → Arbeitsbereich → Sprache = englisch**
   – **Project → Options → Desktop → Language = german**
3. Create a new project via **File → New**.
4. In the ”Target system settings” dialogue, select the respective control system.
5. Accept the default setting.
   A dialogue for the first block is displayed. The name can be freely selected (e.g. PLC_PRG). As a first block, a program in the language ST (structured text) is created in the default setting.



ETCM019

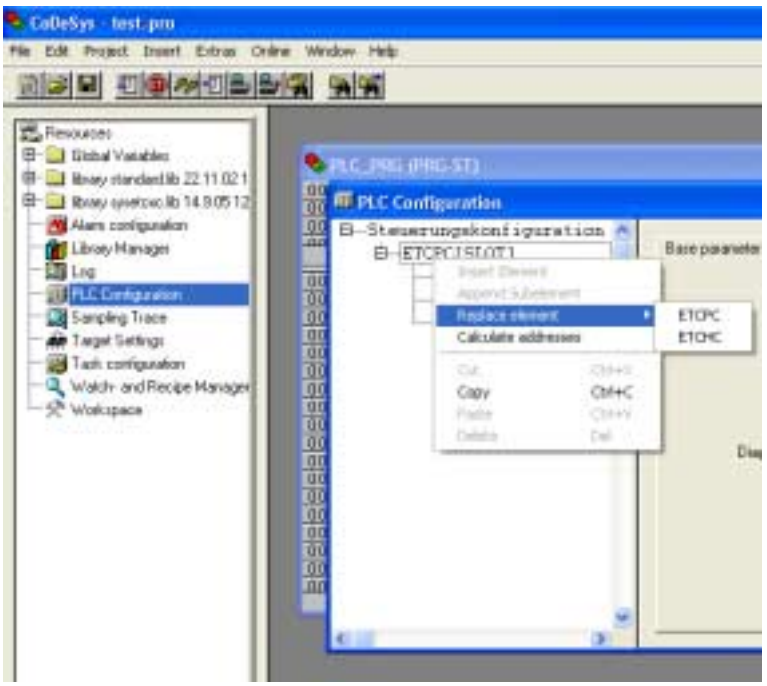
6. Press **<OK>**
7. Save the new file via **File → Save**.



ETCM020

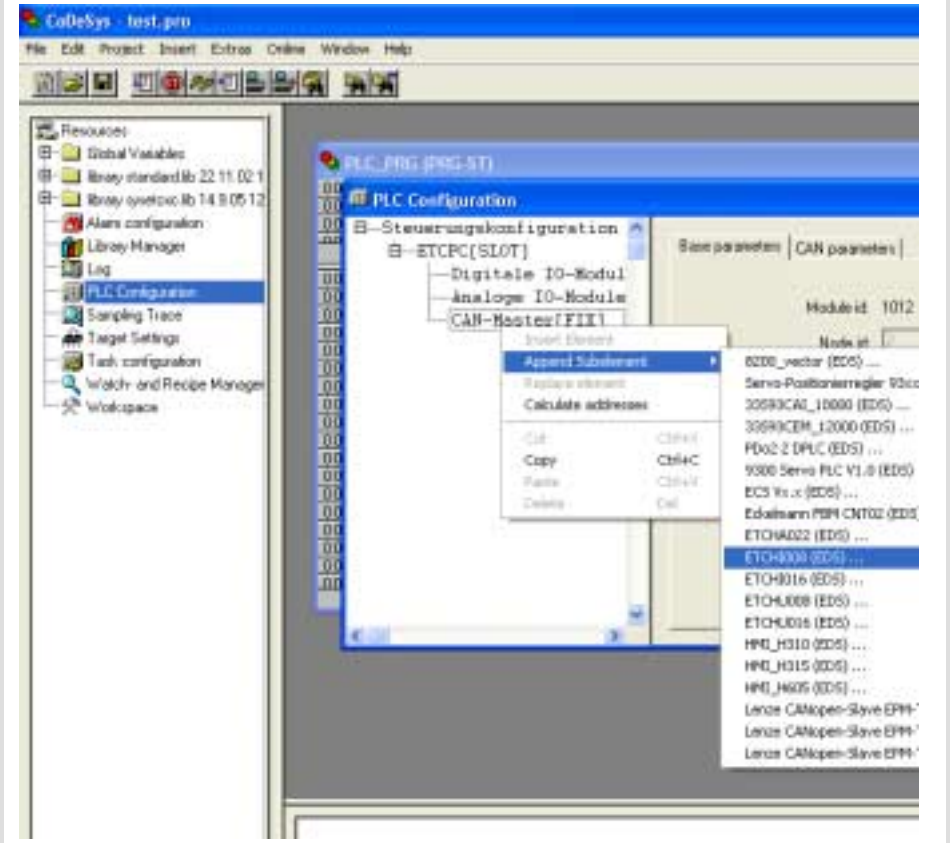
Select control configuration in the CoDeSys

1. In the left lower window area, click the ”Resources” tab.
2. Select ”PLC Configuration”.
3. In the right window area, open the control configuration.
   The entry ”ETCPC [Slot]” is displayed.
4. Right-click ”ETCPC [Slot]”.
5. In the context menu, select the menu item **Replace element → ETHC**.



Configure control system (CAN master)

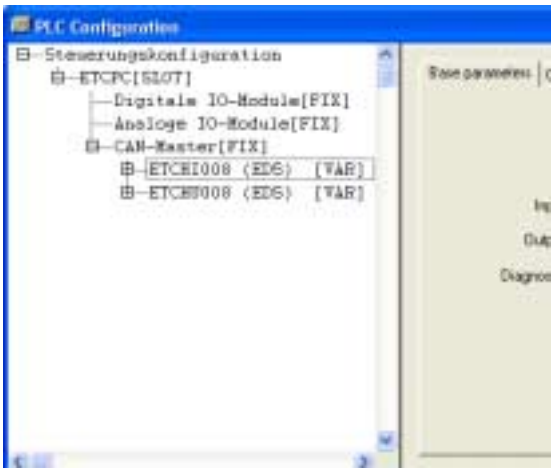
1. Open the entry **ETCHC [Slot]** by clicking ”+”.
2. Right-click on **CAN-Master (Fix)**.
3. In the context menu, select the entry **Append Subelements**. In the submenu, select the module you want to add.



ETCN019

4. Repeat the last step for all modules that you want to add in the order of their position on the DIN rail.

   After the modules have been installed, the configuration looks as follows:



ETCN020

Note!

New modules are added at the end of the list. To insert a new module before an existing one, select the existing module and add it via the right mouse button with **Element einfügen (Insert Element)**.

A module can be removed from the list via the right mouse button and **Delete**.

Description of the module in the EDS file

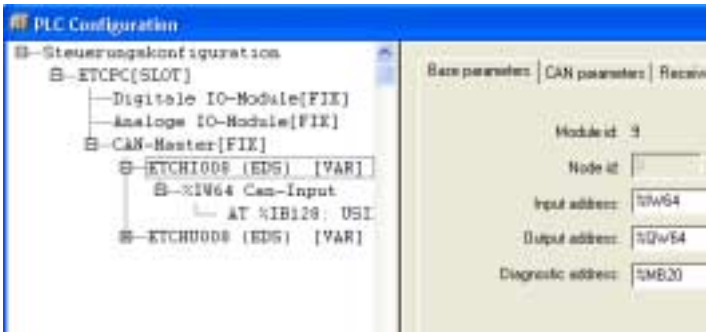
Each module that is configured in the control system requires an EDS file. The EDS file belonging to the module must be located in the target directory of the selected control system (in this case Lenze).

An EDS file contains entries of keywords that describe the property of the module with the respective values.

In the case of modules from third-party manufacturers, the corresponding EDS files must be copied into the target directory of the control system.

Parameterise modules

After you have clicked the individual modules, the display looks as follows:



ETCN021

The program must know at which addresses of the memory area the input and output data of the modules are located. The default setting in the program contains byte addresses (*%IB*, *%QB*), which must be changed into word addresses (*%IW*, *%QW*) because the process image of the ETC is organised according to words. The address range starts at word 64. Proceed as follows to change the addresses into word addresses:

► Starting with the first module in the list (in this case ETCHI008) in the order of the modules (in this case ETCHU008), change the type of the variables *%IB*, *%QB* to *%IW*, *%QW*. The first module receives the address word 64, the following modules receive the word address word 64 plus the byte address minus 128.

Module	Displayed input address	Displayed output address		Input input address	Input output address	Diagnostics address
ETCHI008	%IB128	%QB128	→	%IW64	%QW64	unchanged
ETCHU008	%IB129	%QB128	→	%IW65	%QW64	
Next module	%IB129	%QB130	→	%IW65	%QW66	

It is not necessary to change the diagnostics address.

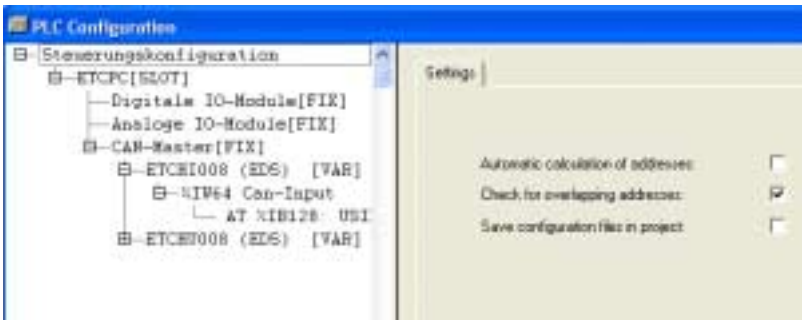
Note!

The described address allocation method prevents the addresses of the individual modules from overlapping. However, the allocation is not optimum in terms of space.

For an optimum assignment of the addresses of the following modules, they must be calculated. For simplification reasons, this is not done here.

Addresses automatically

In the control configuration, the ”Automatic calculation of adresses” checkmark should not be set. Otherwise, the CoDeSys will newly assign the addresses when the control configuration is changed.



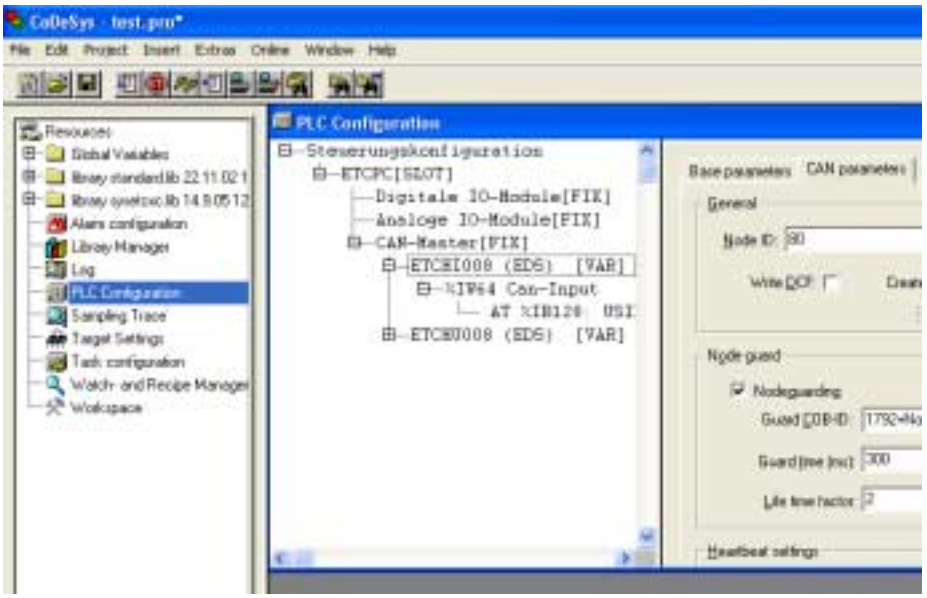
Enter CAN address

Each module has its own CAN address (node ID). An address consists of the type-specific basic address and the individually set address at the hex switch on the front side.

ETCM029

If the HEX switch is set to zero (arrow points to the left) for the fieldbus modules used, the following resulting node IDs must be entered for the individual modules:

Module	CAN basic address	Set at the hex switch	Resulting node ID
ETCHI0xx	80_{dez}	0	80_{dez}
ETCHU0xx	16_{dez}	0	16_{dez}
ETCHA022	48_{dez}	0	48_{dez}



ETCN023

Nodeguarding

If the **Nodeguarding** option is activated, a message is sent to the module at the interval specified in milliseconds under **Guard Time**. If the module does not respond and the number of attempts (**Life Time Factor**) has been reached, the module is regarded as not NOK (defect or not existing).

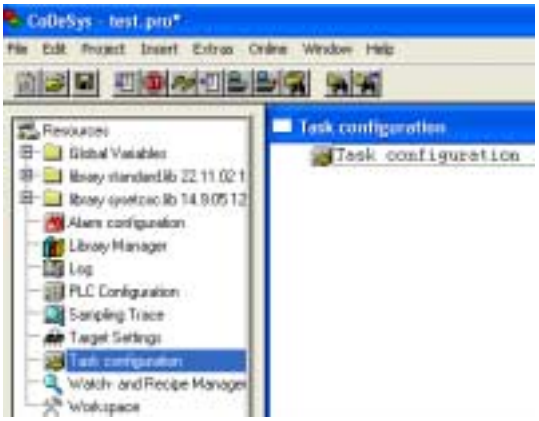
The status of the module is saved in the system variable of the control system DB1_NC2SPS_COPMODUL_VORHANDEN_AW (📖 240).

For all modules, the standard setting is **Guard Time = 500** and **Life Time Factor = 3**. This setting means that a failure is detected after 500 ms x 3 = 1.5 s. If no Guard Time and no Life Time Factor are specified (= 0), the module is not monitored.

Further settings are not required.

Task configuration

A task configuration is only required if system variables of data block_0 are used (📖 342).

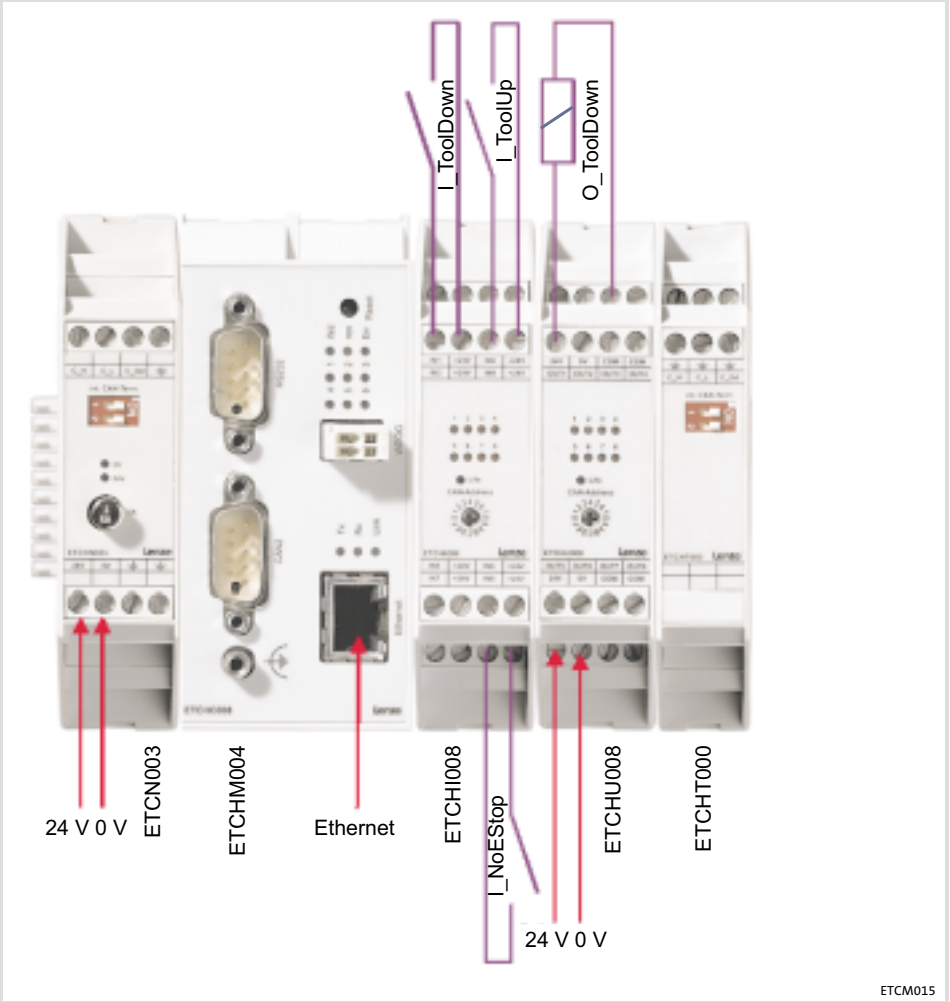


ETCN024

2.12 Creating a PLC sample program

This chapter describes the creation of a PLC program using an ETCHx system as example. Please note the differences in the case of an ETCPx system.

2.12.1 Required hardware



- The 24 V voltage is connected to the modules ETCHN003 (power supply unit) and ETCHU008 (output module).
- A switch **Tool down** (I_ToolDown) is connected at the input IN3 of the ETCHI008.
- A switch **Tool up** (I_ToolUp) is connected at the input IN4 of the ETCHI008.
- A switch **Emergency stop** (I_NoEStop) is connected at the input IN8 of the ETCHI008.
- A relay **Tool down** (O_ToolDown) is connected at the output OUT1 of the ETCHU008.

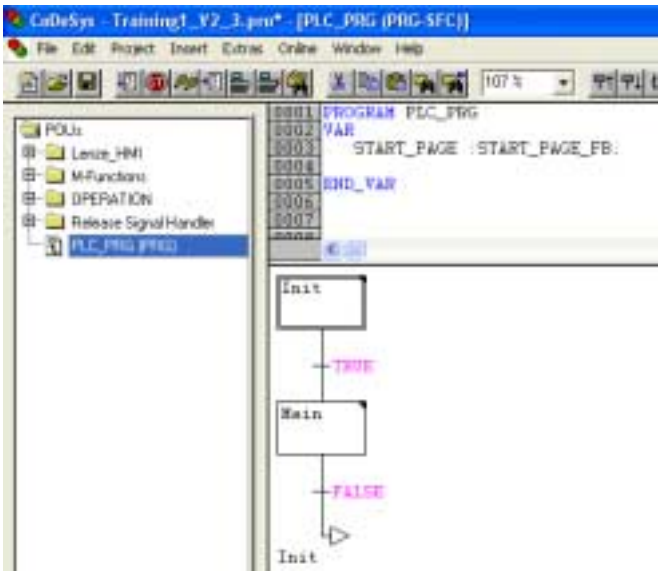
2.12.2 Starting and configuring the PLC sample program

The sample program ”Training1” contains

- ► all settings of the control system,
- ► the parameterisation,
- ► the calls for adding the required PLC functions for the M-functions to the described CNC program.

Start sample program

1. In Windows File Explorer, double-click on the file ”Training1.pro”.
   CoDeSys starts and the sample program is loaded. It consists of a sequence control (AS) with the steps ”Init” and ”Main”.

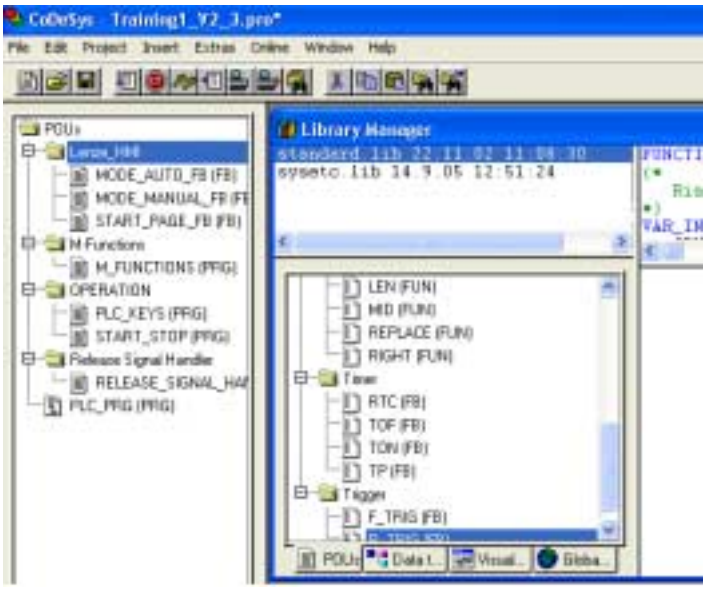


ETCN028

Add libraries

The sample program requires the libraries ”standard.lib” and ”sysetcxc.lib.lib” (...\CoDeSys V2.3\Targets\Lenze\ETCxC).

2. If the libraries do not yet exist, add them via **Resources → Library Manager”**.

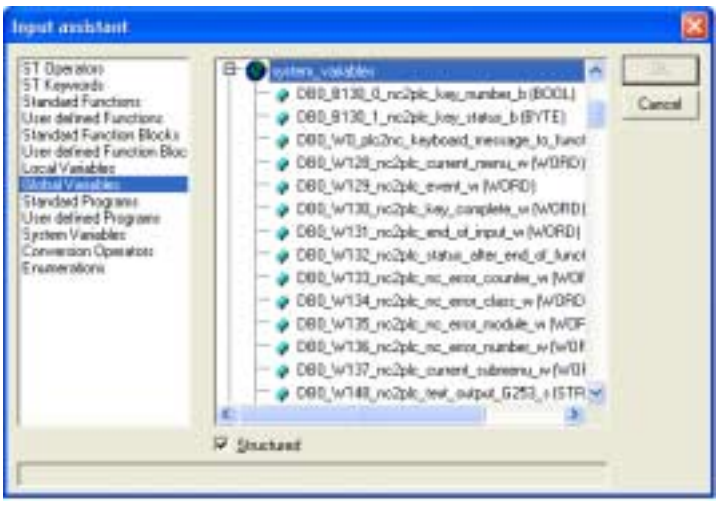


ETCN025

System variables and data blocks

The communication between the CNC control system and the PLC program takes place via variables of the data block 1, the communication between the ETC-MMI and the PLC program via variables of the data block 2. The required system variables are summarised in the following table (the system variables are listed in German and English):

System variable	Flag word	Required for
db1_sps2nc_notaus_bit DB1_X0_0_plc2nc_emergency_bit	%MX1.0.0	Release of CNC
db1_sps2nc_vorschubfreigabe_w DB1_W1_plc2nc_feed_release_axes_w	%MW1.1	Release of CNC
db1_sps2nc_verfahrtastenfreigabe_plus_w DB1_W3_plc2nc_release_jog_keys_plus_w	%MW1.3	Manual travel of keys, e.g. Lenze-HMI, coded bit by bit for 12 drives
db1_sps2nc_verfahrtastenfreigabe_minus_w DB1_W4_plc2nc_release_jog_keys_minus_w	%MW1.4	Manual travel of keys, e.g. Lenze-HMI, coded bit by bit for 12 drives
db1_sps2nc_reglerfreigabe_w DB1_W5_plc2nc_regulator_release_w	%MW1.5	Release of CNC
db1_sps2nc_einlesefreigabe_bit DB1_X7_0_plc2nc_read_in_release_bit	%MX1.7.0	Release of CNC
db1_sps2nc_programmstart_b DB1_B9_0_plc2nc_program_start_b	%MB1.9.0	ETC-MMI
db1_sps2nc_programmstop_b DB1_B12_0_plc2nc_stop_program_b	%MB1.12.0	ETC-MMI
db1_nc2sps_betriebsbereit_2_bit DB1_X128_0_nc2plc_NC_ready_1_bit	%MX1.128.1	Monitoring CNC
db1_nc2sps_programmstart_b DB1_B137_0_nc2plc_program_start_b	%MB1.137.0	ETC-MMI
db1_nc2sps_programmstop_b DB1_B142_0_nc2plc_stop_program_b	%MB1.142.0	ETC-MMI



ETCN026

Note!

The complete list of the system variables can be displayed in the programming window of the ETC-CoDeSys via **<F2>**.

Task for M14 and M15

In a PLC program, the two functions M14 and M15 are to be programmed as follows:

M-function	Target function	Programming in the PLC program
M15	Lift tool	Output ”tool down” (O_ToolDown) = 0 Wait for ”tool up” (I_ToolUp) = 1
M14	Lower tool	Output ”tool down” (O_ToolDown) = 1 Wait for ”tool up” (I_ToolDown) = 1
M30	Program end	--

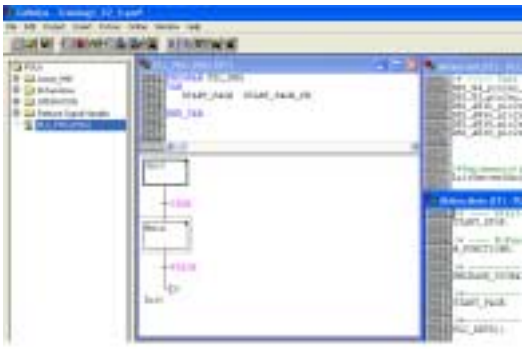
Sequence control

A program in the language ”sequence control” consists of the basic elements ”step”, ”transition” and ”branch”. In the sequential function chart, elements can be added via the right mouse button.

The program ”Training1” consists of an initialisation block with two steps and one transition.

Element	Name	Contents	Comment
Step	Init	Initialisation of the system variables	Initialisation program
Transition		True	Switchover tot he next step ”Main”.
Step	Main	Block START_STOP	Evaluation of the system variables for Start and Stop.
		Block M_FUNCTIONS	Processing of the M-functions
		Block RELEASE_SIGNAL_HANDLER	Release of the system variables, evaluation of the input ”emergency stop” (I_NoEStop).
		Block PLC_KEYS	Execution of the PLC keys of the ETC-MMI and the M-functions M14 and M15.
Transition		False	No switchover to the next step due to FALSE. The program remains in step ”Main”.

After you have double-clicked on a step, a window opens where the corresponding action of the step is entered.



ETCN029

Define inputs/outputs

1. In the CoDeSys, select **Resources → Global Variables → Ein_Ausgaenge**.
2. Define the inputs and outputs as bit in the corresponding words of the process image.

ETCN030

Function START_STOP

This function evaluates the following system variables, which control start and stop of the PLC program in the ETC-MMI:

- db1_sps2nc_programmstart_b
- db1_sps2nc_programmstop_b
- db1_nc2sps_programmstart_b
- db1_nc2sps_programmstop_b
- (DB1_B9_0_plc2nc_program_start_b,
  DB1_B137_0_nc2plc_program_start_b;
  DB1_B12_0_plc2nc_stop_program_b,
  DB1_B142_0_nc2plc_stop_program_b)

Function RELEASE_SIGNAL_HANDLER

This function evaluates the following system variables, which set the release and monitoring signals in the control system:

- db1_sps2nc_vorschubfreigabe_w
- db1_sps2nc_reglerfreigabe_w
- db1_nc2sps_betriebsbereit_2_bit
- DB1_W1_plc2nc_feed_release_axes_w
- DB1_W5_plc2nc_regulator_release_w
- DB1_X128_1_nc2plc_NC_ready_2_bit

Function M_FUNCTIONS

The function block M_FUNCTIONS evaluates the M-functions coming from the NC. By means of the two following system variables, the block detects whether a new M-function has been requested:

- DB1_NC2SPS_MFKT_STROBE_BIT, DB1_SPS2NC_MFKT_QUITT_BIT
- (DB1_X150_0_nc2plc_M_function_strobe_bit, DB1_X32_0_plc2nc_acknowledgement_for_M_function_bit)

Via the following system variable, the block detects which M-function has been requested:

- DB1_NC2SPS_MFKT_W (DB1_W151_nc2plc_M_function_w)

As long as an M-function is active, a window opens in the ETC-MMI via the bit DB2_HINT_OPTIONALLYSTOP_ACTIVE_BIT in the system variable DB2_SPS2HMI_HINWEIS_AW.

ETCN031

After it has been detected that the new M-function is M14 or M15, the M-function is processed in the program PLC_KEYS.

M14

The output *O_ToolDown* is set and a timer of 5 s is started. If the input *I_ToolDown* is set within the next 5 s, the function is switched, otherwise an error message is displayed. The error message is realised via the function SPSERROR, which opens an error window in the ETC-MMI via an error number (*F_TOOL_UP_KI*).

```
(*Implementation of the PLC-keys of the ETC-KBI *)
(*F1: DRIVES ON/OFF (in release signal handler) *)
(*F4: Tool open/close                           *)

(* tool down *)
SR_Tool_down(SET1:=MMI_PLC_key_tool_down);

TRIG_M14(CLK:=M_M14_bit OR SR_Tool_down.Q1);
IF (M_M14_bit OR SR_Tool_down.Q1) THEN
   IF (TRIG_M14.Q) THEN
      TIMER_M14(IN:=FALSE);
      SR_Tool_down.RESET:=FALSE;
      O_ToolDown := TRUE;
   ELSE
      TIMER_M14(IN:=TRUE, PT:=#5s);
      IF (I_ToolDown OR Tool_down_q) THEN
         M_M14_bit := FALSE;
         SR_Tool_down.RESET:=TRUE;
      ELSIF (TIMER_M14.Q) THEN
         M_M14_bit := FALSE;
         SR_Tool_down.RESET:=TRUE;
         SPSERROR(F_TOOL_UP_KI, 3, '', 0);
      END_IF
   END_IF
END_IF
```

ETCN032

M15

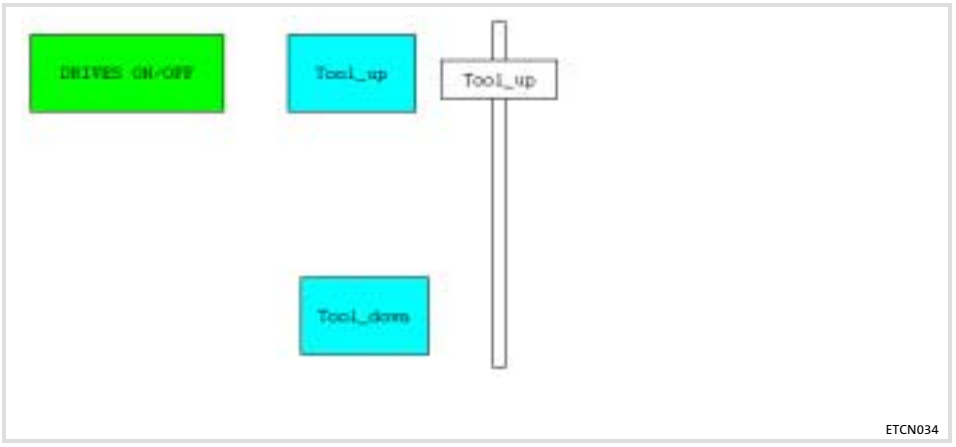
The output *O_ToolDown* is reset and a timer of 5 s is started. If the input *I_ToolUp* is set within the next 5<s, the function is switched, otherwise an error message is displayed.

```
(* tool up *)
SR_Tool_up(SET1:=MMI_PLC_key_tool_up);

TRIG_M15(CLK:=M_M15_bit OR SR_Tool_up.Q1);
IF (M_M15_bit OR SR_Tool_up.Q1) THEN
   IF (TRIG_M15.Q) THEN
      TIMER_M15(IN:=FALSE);
      SR_Tool_up.RESET:=FALSE;
      O_ToolDown := FALSE;
   ELSE
      TIMER_M15(IN:=TRUE, PT:=t#5s);
      IF (I_ToolUp OR Tool_up_g) THEN
         M_M15_bit := FALSE;
         SR_Tool_up.RESET:=TRUE;
      ELSIF (TIMER_M15.Q) THEN
         M_M15_bit := FALSE;
         SR_Tool_up.RESET:=TRUE;
         SPSPRAC9(P_TOOT_DOWN_KT, 3, '', 0);
      END_IF
   END_IF
END_IF;
```

ETCN033

If no I/O modules are connected to the control system, the visualisation can be used.



ETCN034

”DRIVES ON/OFF” switches the drives on and off. ”Tool_up” and ”Tool_down” are the switches that are to be activated.

2.12.3 Loading the PLC sample program into the control system

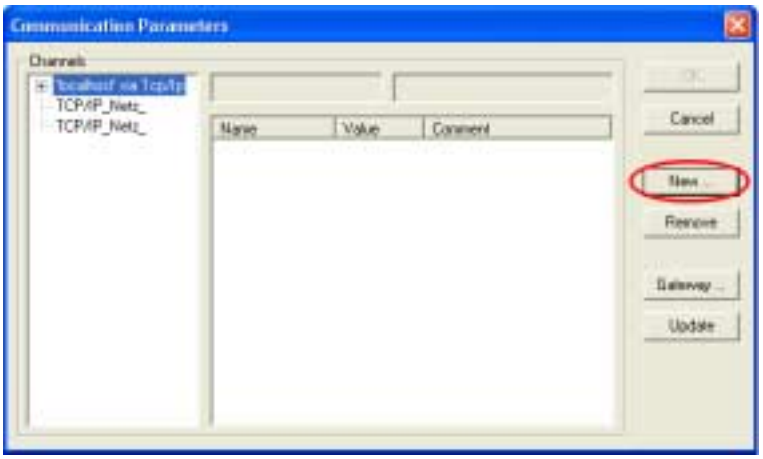
Note!

The steps described in this chapter only apply to the ETCHx variant (DIN rail variant); they are not required for the ETCPx variant (PCI card).

Via the ETC-CoDeSys, the connection from the PC to the control system can be set, optionally via a serial connection (RS232 interface) or via the network.

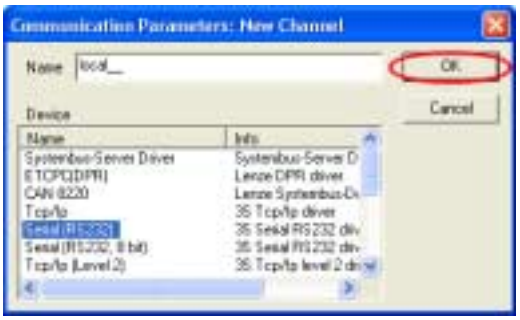
Serial connection

1. Connect the PC and ETC via the serial interface.
2. Select **Online → Communication Parameters**.
3. Click on **New**.



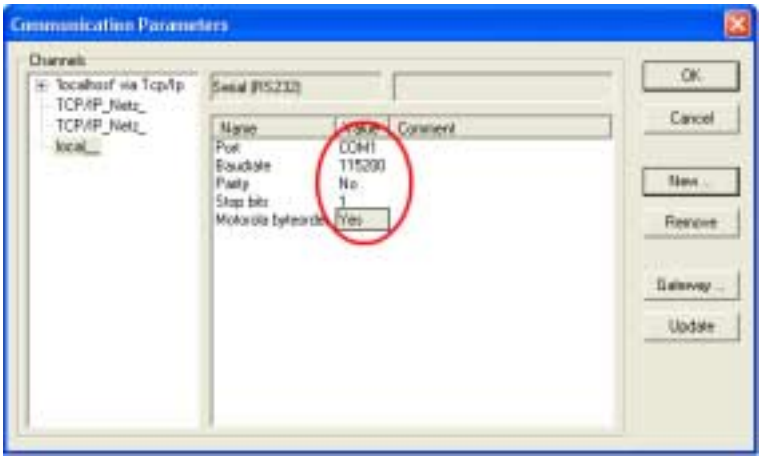
ETCM036

4. Enter a connection name (e.g. local_) and select the serial driver "Serial RS232".



ETCM037

5. Click **OK**.
6. Specify the parameters of the interface (e.g. COM1 with baud rate, parity and stop bit). For this purpose, double-click the corresponding field.

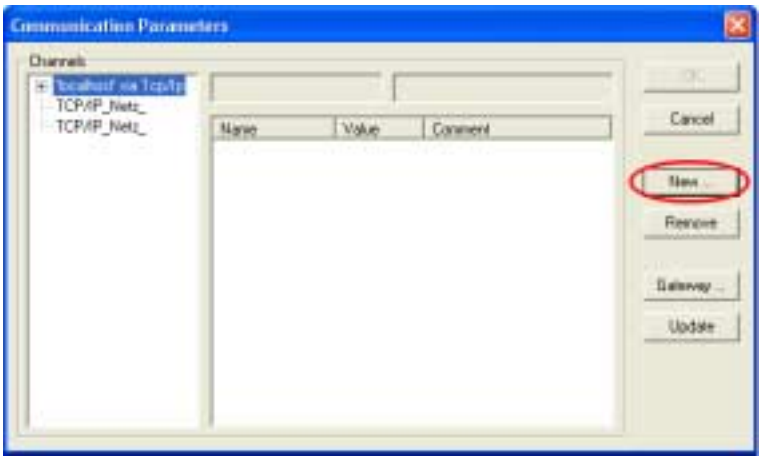


ETCM038

7. As the control system contains a Motorola processor, the field ”Motorola byteorder” must be set to ”Yes”. If required, correct this setting.
8. Close the window with **OK**.

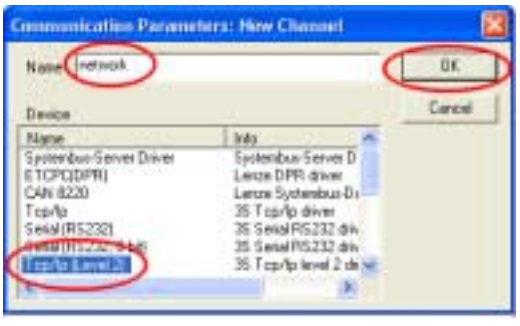
Connection via network

1. Select **Online → Communication parameters**.
2. Click on **New**.



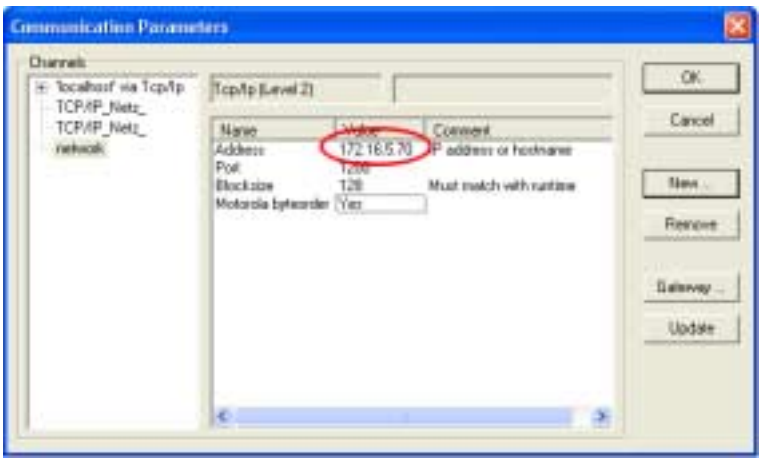
ETCM036

3. Specify a name (e.g. network) and select the TCP/IP protocol as parameter. Click **OK**.



ETCM040

4. Double-click the ”Address” field and specify the correct IP address. (📖 28).



ETCM041

Note!

Do not use any leading zeros in the IP addresses. Otherwise, the IP address will be interpreted as octa decimal number.

5. As the control system contains a Motorola processor, the field ”Motorola byteorder” must be set to ”Yes”. If required, correct this setting by double-clicking several times.

Note!

The left window area shows the connection options, in this case the two connections ”local_” and ”network”.

The current setting is the setting that is displayed in the right window before you click **OK**, it is used for log in.

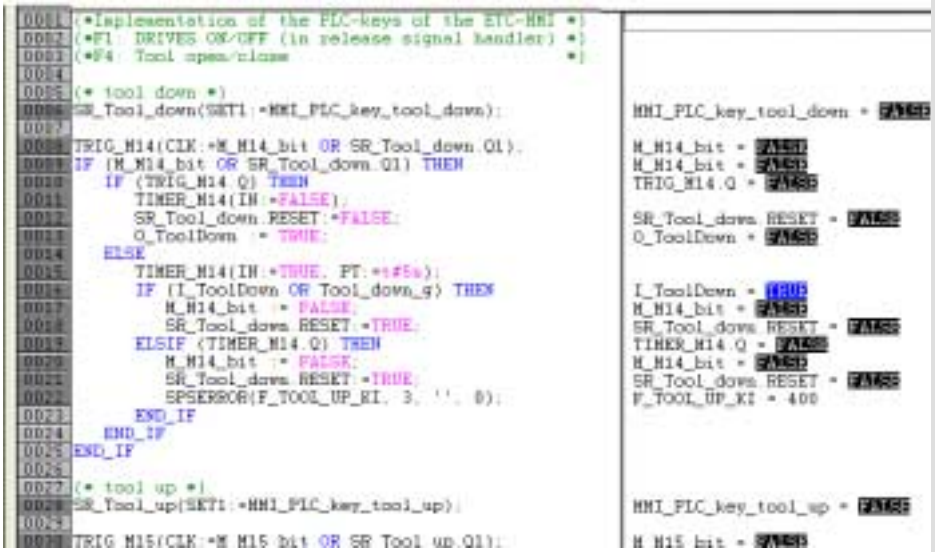
6. Close the window with **OK**.

Load and start program

1. Select **Online → Log in**.
   The program is loaded into the control system.
2. Select **Online → Start**.
   The program starts. After a successful start, the ”RUNNING” field in the status line changes its colour from grey to black.

ETCM042

The status line shows the current connection (in this case: ”local_”). In the right part of the window, the program displays the current values of the variables.



ETCNC097

Operate program

After you have logged in to the control system via **Online → Log in**, the following functions can be used:

- Stop program: **Online → Stop**
- Reset data memory to starting values: **Online → Reset**
- Create boot project: **Online → Create boot project**
  The program is saved as ”default.prg” in the flash of the control system. When the control system is started, it looks for a program ”default.prg”. This program is automatically started as boot project.
- Delete boot project: **Online → Reset**
- Stop communication with the control system: **Online → Log out**
  The program in the control system remains in the status that was selected last.

2.13 Testing CNC and PLC program

Note!

Basically, the operation of the CNC program is possible:

- ► without connected mechanics and drives.
  This is achieved by setting the machine constant MK_TEST_OHNEMECHANIK=1.
- ► without a PLC program.
  This is achieved by setting the machine constant MK_SPS_DUMMY=1.

For the following tests with program and mechanics, the machine constants must be set as follows:

Test setting

MC keyword	Number of values	Values
MK_TEST_OHNEMECHANIK	1	0
MK_SPS_DUMMY	1	0

Connecting the hardware

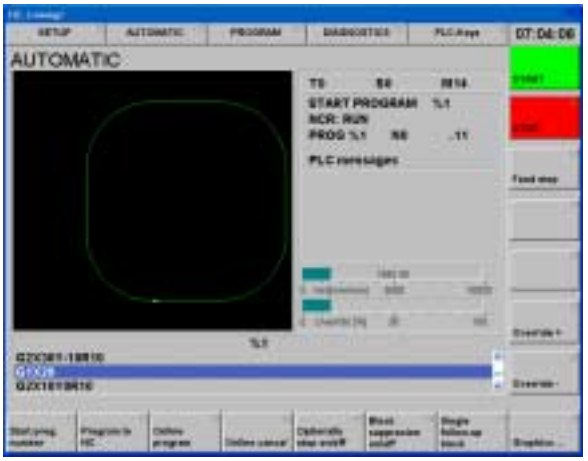
1. Connect the hardware according to the requirements (📖 60).
   – The inputs ”Tool down” and ”Tool up” must be set according to the progress of the program (M14, M15). Thus, 24 V must be continuously applied to the inputs or the inputs must be switched in time.

Start PLC program

1. Start the PLC sample program in the CoDeSys. For this purpose, double-click on the file ”Training1.pro” in Windows File Explorer (📖 61).
   CoDeSys starts and the sample program is loaded.
2. Load the PLC program into the control system via **Online → Log in** (📖 68).
3. Start the PLC program via **Online → Start** (📖 68).

Start CNC program

1. Start the ETC-MMI via **<Start> → Programs → Lenze → ETC**
   In the ETC-MMI window, press **<F11>** (program).
   The MMI window opens in the ”Programming” operating mode. It shows the text editor for the CNC program.
2. Press **<F8>** (Graphics) and afterwards **<F7>**(Graphics on/off).
   The window is split. The text editor is displayed on the left, the display field for the graph (graphic area) on the right.
3. Press **<F2>** (Open program).
4. Select the CNC program (e.g. **test.din**) and press **<Enter>**
5. Press **<F7>** (Program to NC).
6. Select the CNC program (e.g. **test.din**) and press **<Enter>**
   The CNC program is transferred to the control system (📖 49).
7. Press **<S1>** (Start).
   The drives start and the profile of the rectangle are executed (📖 49).
   The drawn straight line from the zero point (0/0) to the starting point (10/10) is now executed when the drive is also in the zero point. Otherwise, the control system traverses the drives on a straight line from the current point to the starting point.
8. Change the material speed to a small value (e.g. 1 %). For this purpose, click **<S7>**(Override -).
   When the PLC and CNC programs are loaded correctly, a marker follows the circular profile. The output ”Tool down” is set and reset according to the called M-function.

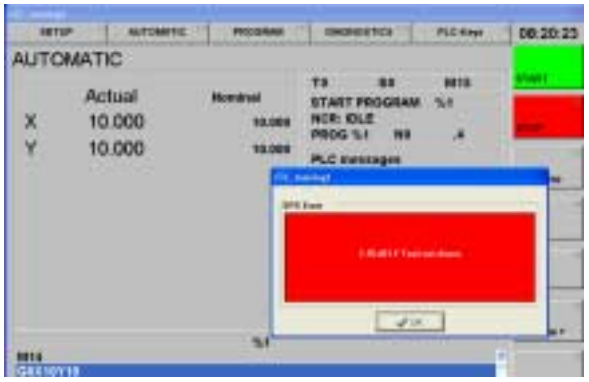


ETCN035

9. To stop the drives, press **<S2>** (Stop).

Error messages of the CNC program

If 24 V is not applied to one of the two inputs in time, the program stops and the following error message appears in a red box.



ETCN036

This error message is generated via the **SPSERROR** function in the **PLC_KEYS** module after a timer of 5 s has elapsed. The text belonging to the error number is contained in the file ”sps_fehl.db” (English version: ”sps_erro.db”). This file is located in the directory ”cfg” of the ETC-MMI.

The figure shows an error from the control system (2) of the PLC program module (10) with the error number 400 (F_TOOL_UP_KI). For a correct error message, an error number 2.10.400 with the corresponding text must be entered in the file ”sps_fehl.db”.

2.14 PLC keys in the ETC-MMI

In the ”PLC KEYS” operating mode, it is possible to start manual functions in the PLC from the ETC-MMI. For this purpose, 6 menu levels are available. This corresponds to 42 function keys.

In this example, the following assignments are to be made to three PLC keys:

F1: System on/off

F4: Lower tool

F5: Lift tool

2.14.1 Labelling of the PLC keys in the ETC-MMI

The labelling of the PLC KEYS is realised in the language-dependent file **lenze.txt**.

```
;Dynamic PLC key assignments
00002799,"PLC-Keys@C126322560@Arial,8,1,0,0,15359"
00002800,"Power on [00]@C126322560@Arial,8,1,0,0,6"
00002801,"Power off [01]@C126322560@Arial,8,1,0,0,6"
00002802,"Drives on [02]@C126322560@Arial,8,1,0,0,6"
00002803,"Drives off [03]@C126322560@Arial,8,1,0,0,6"
00002804,"Valve tool clamp [04]@C126322560@Arial,8,1,0,0,6"
00002805,"Text [05]@C126322560@Arial,8,1,0,0,6"
00002806,"Text [06]@C126322560@Arial,8,1,0,0,6"
00002807,"Text [07]@C126322560@Arial,8,1,0,0,6"
00002808,"Text [08]@C126322560@Arial,8,1,0,0,6"
00002809,"Text [09]@C126322560@Arial,8,1,0,0,6"
00002810,"Text [10]@C126322560@Arial,8,1,0,0,6"
00002811,"Text [11]@C126322560@Arial,8,1,0,0,6"
00002812,"Text [12]@C126322560@Arial,8,1,0,0,6"
00002813,"Text [13]@C126322560@Arial,8,1,0,0,6"
```

ETCN037

These texts are then assigned to the keys by the configuration file **delphmmi.ini** :

```
[SPS-Tasten]
Keycode=Shift-F8
;Key00 bis Key06 auf erster Seite SPS-Tasten
;Key07 bis Key13 auf zweiter Seite SPS-Tasten
;Key14 bis Key15 auf dritter Seite SPS-Tasten
;...usw.
Key00=2800,2801
Key01=2802,2803
Key02=2804,2805
Key03=2806
Key04=2807
Key05=-1
Key06=-1
Key07=2820,2821
Key08=2822,2823
Key09=2824,2825
Key10=2826,2827
Key11=2828,2829
Key12=2830,2831
Key13=2832,2833
Key14=-1
Key15=-1
Key16=-1
Key17=-1
```

ETCN038

Key 00: Texts 2800 and 2801 from the file lenze.txt are used for the first PLC key.

2 texts for one key automatically contain a toggle function. This means that the system is switched on when F1 is pressed and switched off when F1 is pressed again. The output value in the PLC is switched over every time the key is pressed.

Key 03: Text 2805 (Lower tool) of the file ”lenze.txt” applies.

Key 04: Text 2806 (Lift tool) of the file ”lenze.txt” applies.

Representation in the ETC-MMI:

ETCN039

2.14.2 Calling the signals in the PLC

ETCN040

In the example, an absolute assignment is made to the word variable:

MMI_PLC_key_DRIVES_ON → %MX1.121.0 (X=bit) accesses bit 0 of the DW212 in the DB1.

Access to the symbolic system variable is also possible:

DB1_AW212_mmi2plc_jog_key_aw [0].0

DB2_MMI2SPS_TASTEN_AW [0].0

2.14.3 Function of the keys

The keys in the ETC-MMI have the same function as the M-functions M14 and M15; i.e.they can be used to lower and lift the tool. The input or the signal of the visualisation must be applied within 5 s. Otherwise, an error message appears.

2.15 Operation via a Lenze-HMI

In addition to operation via the ETC-MMI (with PC), operation via a Lenze-HMI is also possible.

The HIM, which is connected via the CAN1 bus, can be used as additional or main operating control.

If the HMI is used for the complete machine control, the control system must be parameterised and programmed by means of a PC first.

In this example, the HMI505 is used as additional operating control in addition to the ETC-MMI.

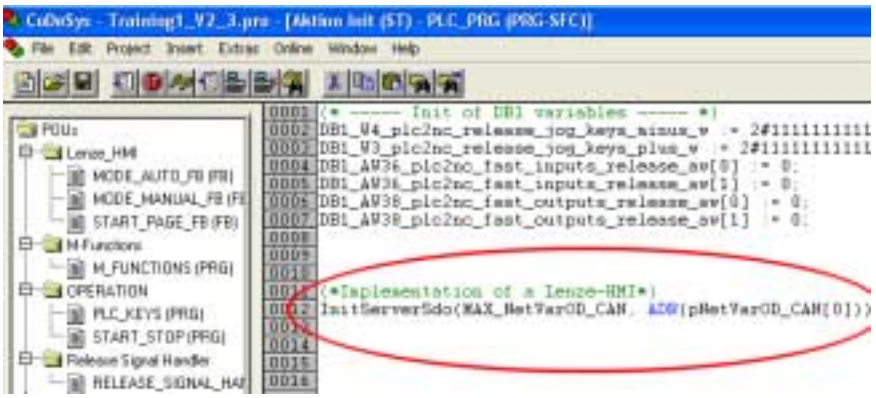
In addition to the manual travel function, the NC program %1 can be started from the HMI505. This function is explained on the basis of a simple HMI project.

The HMI505 project is located on the ETC-CODeSys CD in the HMI designer file **Training1_HMI505_V1.VTS** .

2.15.1 Settings for the connection of a Lenze-HMI H505

Integrate libraries

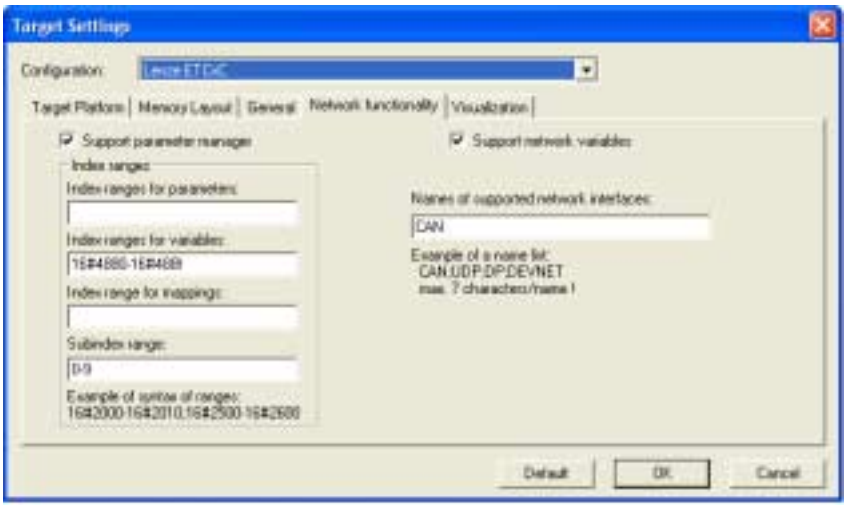
Activation of the communication function in the initialisation step:



To use this function, the library ”ServerSDO.lib” must be integrated. In the process, other libraries are also integrated. These libraries provide the interface functions from the ETC to the HMI:

Settings in the PLC program (target system)

To inform the control system that communication to an HMI is desired, change the system setting as follows:



ETCN043

In the system settings, activate the object directory and the network variables on the ”Network functions” tab.

The object directory provides the declared variables with the corresponding indexes to the HMI (Lenze code = index with CANopen).

The desired number of indexes can be set in the index area. The CAN network acts as network interface to the HMI.

After these settings have been made, the CoDeSys provides the object directory:

ETCN044

CoDeSys object directory

The object directory is the interface between the codes in the HMI H505 and the HEX indexes in the ETC. According to the Lenze standard, the following relationship exists:

INDEX =DEZ_TO_HEX(24575 code)

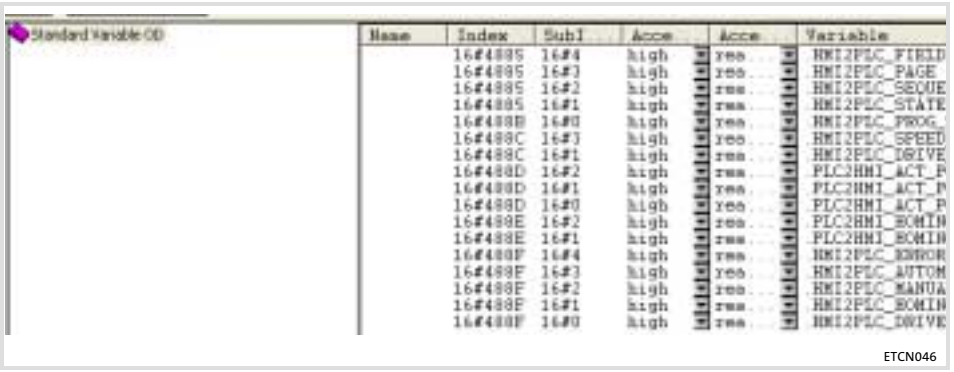
To facilitate this conversion, an Excel table is available (Umrechnung Objektverzeichnis Schulung.xls). It is located on the ETC-CODeSys CD in the directory "Systemhandbuch". In this file, the declaration and the distribution of the HMI variables to the codes can be made.

Lenze Codestelle	Sub-code	HEX-Index	HEX Index	Sub-index	Zugriff	Attribut	Variable
6010	4	4885	16#4885	4	high	read-write	.HMI2PLC_FIELDINDEX
6010	3	4885	16#4885	3	high	read-write	.HMI2PLC_PAGE
6010	2	4885	16#4885	2	high	read-write	.HMI2PLC_SEQUENCE
6010	1	4885	16#4885	1	high	read-write	.HMI2PLC_STATE
6004	0	488B	16#488B	0	high	read-write	.HMI2PLC_PROG_START
6003	3	488C	16#488C	3	high	read-write	.HMI2PLC_SPEED_MANUAL
6003	1	488C	16#488C	1	high	read-write	.HMI2PLC_DRIVE_MANUAL
6002	3	488D	16#488D	3	high	read-write	.PLC2HMI_ACT_POS_Z
6002	2	488D	16#488D	2	high	read-write	.PLC2HMI_ACT_POS_Y
6002	0	488D	16#488D	0	high	read-write	.PLC2HMI_ACT_POS_X
6001	2	488E	16#488E	2	high	read-write	.PLC2HMI_HOMING_OK
6001	1	488E	16#488E	1	high	read-write	.PLC2HMI_HOMING_RUN
6000	4	488F	16#488F	4	high	read-write	.HMI2PLC_ERROR_RESET
6000	3	488F	16#488F	3	high	read-write	.HMI2PLC_AUTOMATIC
6000	2	488F	16#488F	2	high	read-write	.HMI2PLC_MANUAL
6000	1	488F	16#488F	1	high	read-write	.HMI2PLC_HOMING
6000	0	488F	16#488F	0	high	read-write	.HMI2PLC_DRIVES_ON_OFF

ETCN045

The part highlighted in yellow can be copied completely to the object directory of CoDeSys by means of **<Ctrl>+<C>**.

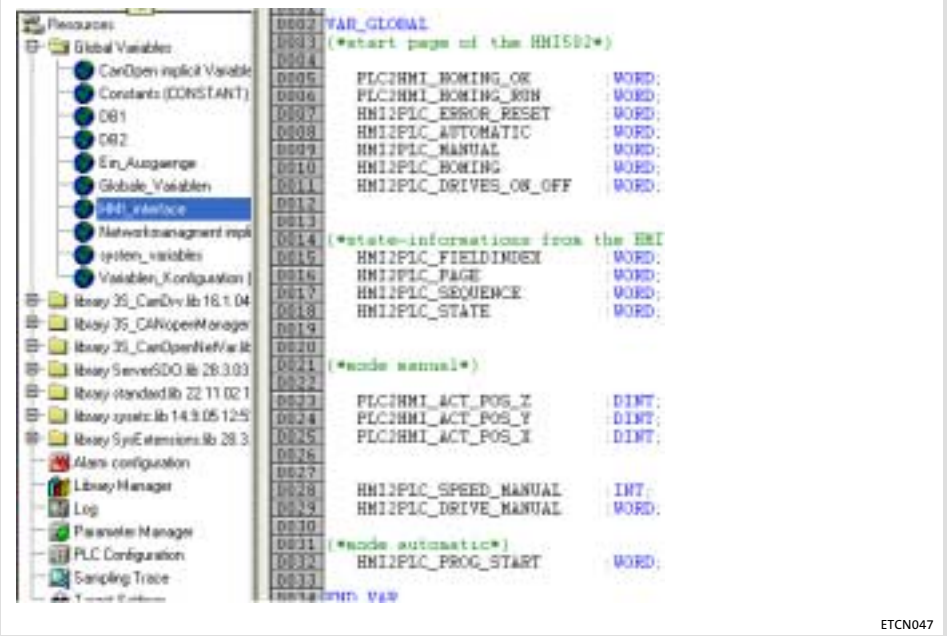
Object directory (Variable tab):



ETCN046

Please pay special attention to the dot before the variable.

The declared variables must be created as global variables:



If the HMI accesses variables that have not been created in the CoDeSys, a communication error occurs.

2.15.2 Functional description HMI505 operation

In the following, the functional description of the project ”ETC-Schulung des HMI” (”ETC training of the HMI”) will be briefly explained.

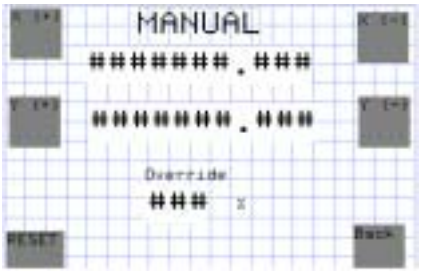
Start page

- ► Selection of the Hand (Manual) and Auto operating modes
- ► Switching the drives on and off
- ► Error reset
- ► Activation of the download page

ETCN048

Manual operating mode

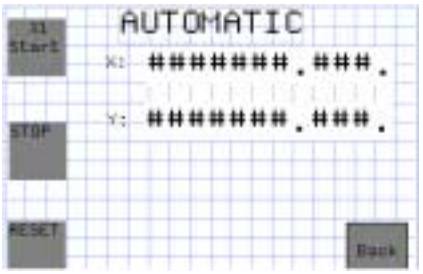
- ► Setting the override for manually traversing the axes
- ► Traversing the X and Y axes in both directions
- ► Error reset
- ► Return to start page
- ► Display of the axis positions



ETCN049

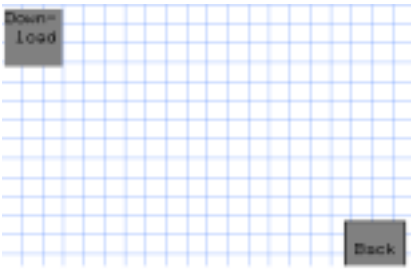
Auto operating mode

- ▶ Error reset
- ▶ Return to start page
- ▶ Display of the axis positions
- ▶ Start of the NC program %1
- ▶ Stop of the running NC program



ETCN056

Download page



ETCN057

Project name: **Schulung1_HMI505_V1.VTS**

2.16 Updating the firmware of the ETCHx in the ”Standalone” operating mode

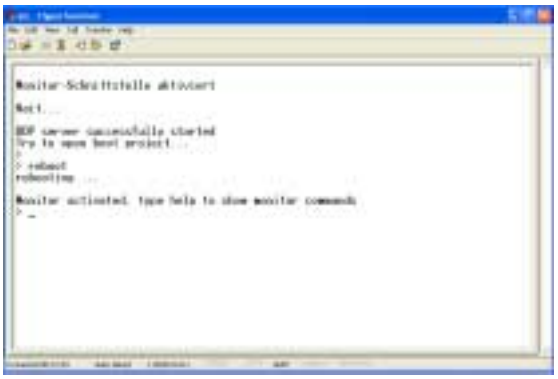
Note!

The steps described in this chapter only apply to the ETCHx variant (DIN rail variant); they are not required for the ETCPx variant (PCI card).

2.16.1 Calling the boot monitor in the control system

Initial state: The communication between PC and ETCHx has been established and the monitor interface activated (📖 21)

1. In the terminal program, enter the command **reboot**.
2. During the running reboot process, press and hold the **<Shift> + <!>** keys.



ETCM012

When the prompt ”>” appears, the input in the boot monitor of the control system is active. On the control system, the LEDs 1 ... 6 flash circulatingly.

Note!

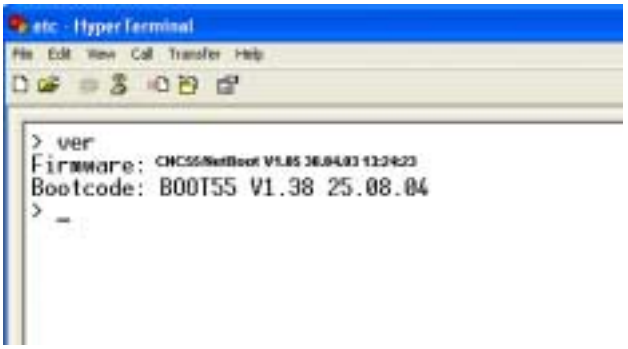
When the firmware is running, the control system can also be reset by switching the control system on/off or by pressing the reset key.

2.16.2 Querying the version of the firmware

1. When the prompt ”>” of the boot monitor is displayed, enter the command **ver**.

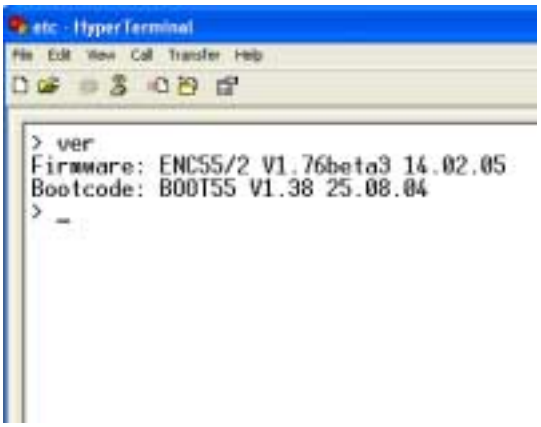
   The version of the firmware is queried and displayed.

Variant ”with ETC-MMI”



ETCN005

Variant ”Standalone”

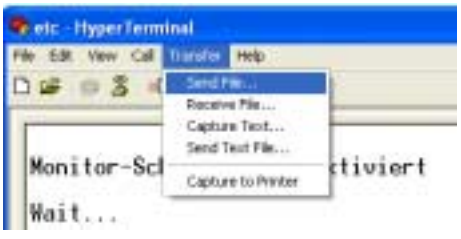


ETCN006

2.16.3 Updating the firmware

Initial state: The communication between PC and ETC has been established and the monitor interface has been activated (📖 21).

1. To transfer the current firmware to the ETC, activate the menu item **Transfer → Send File** in the HyperTerminal.



ETCM021

2. Specify the path to the firmware file and select the protocol ”Zmodem mit Wiederherstellung” (”Zmodem with Crash Recovery”).

Firmware file names: ETCHC_A.rsc (ETCHC with 4 MB) or ETCPC.rsc (ETCPC)

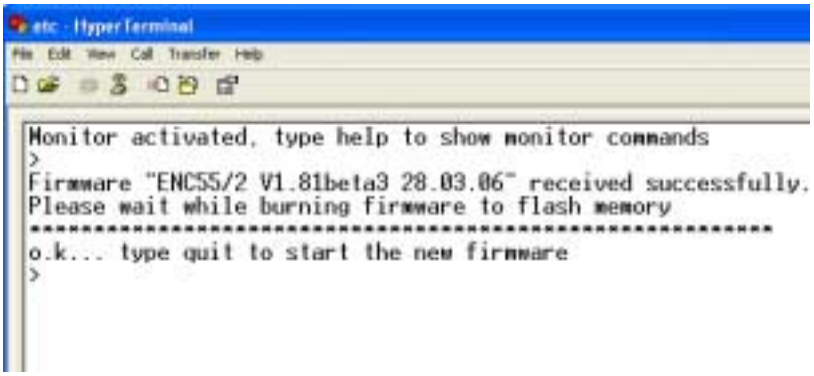
ETCN098

3. Click on **Send**.

   While the file is being loaded into the control system, the following figure is displayed. On the ETC front plate, the LEDs 1 ... 6 flash circulatingly.

ETCN099

After loading is complete, the following figure is displayed:


etc - HyperTerminal
File Edit View Call Transfer Help
Monitor activated, type help to show monitor commands
>
Firmware "ENC55/2 V1.81beta3 28.03.06" received successfully.
Please wait while burning firmware to flash memory
****
o.k... type quit to start the new firmware
>

ETCN100

The control system waits until the firmware is loaded; on the ETC front plate, the LEDs 1 ... 6 light up circulatingly.

4. In the HyperTerminal window, enter **quit** .

   The firmware starts. On the ETC front plate, the green watchdog LED lights up.

Important commands of the boot monitor

Command	Meaning
ver	Shows the versions of the boot loader and the currently loaded firmware.
sz [file name]	Sends the currently loaded firmware under the specified name to the PC via Z modem.
rz	When ”Dateien senden” (”Transmit files”) is selected, this command is sent implicitly by the connected terminal program to the control system. It is used for the transmission of firmware or boot loader to the control system. For the transmission, the ”ZMODEM” protocol must be selected.
quit	Quits the boot monitor and starts the loaded firmware.
reset	Resets the control system.
help	Shows the commands of the boot monitor.

3 CNC programming

This chapter describes the functions of the ETCxC control, with which the programs for processing workpieces are created. The way ETCxC is programmed is based on DIN 66025. Compared to the DIN, the instruction set is provided with a number of additional, better performing functions.

In the following, first the some basic conditions for the creation of a program are described. Then all the functions are described in detail.

3.1 Basics

Program creation variants

The programs can be created either on the control itself (📖 47) or externally, using any ASCII Editor, in accordance with the following rules.

Program transfer

If the control is connected to an ETC-MMI via Ethernet interface, the programs are directly transferred from the hard disk of the PC to the control when the control is started.

If no ETC-MMI is used as a console, the programs can be transferred via the monitor interface (serial interface) of the control by means of modem. Programs which are saved in the control can then also be read out via the monitor interface.

To transfer programs via the monitor interface or a disk to the control, these must have a number consisting of up to 4 digits and the extension **.DIN** as the name. The number is automatically used as the program number and inserted with a preceding % sign at the start of the program. The programs are automatically saved in the EEPROM of the control if the machine constant is MK_NCPROG_NICHT_INS_EEPROM = 0.

If an ETC-MMI is used, it is also possible to process online programs. Programs of this kind can have any length and are processed during the transfer. Online programs must be created sequentially and must not contain any jumps or other branches. Subprogram calls with “G22 Lxxx” are allowed.

Program components

A program consists of a program number (program start), a number of blocks and the program end. It must consist of at least the program start and the program end. A program can contain any number of blocks. The only limit is placed by the available total memory. The number of programs is limited to an adjustable value (📖 200).

No control characters must be used in a program.

Program start

Generally, a program starts with a ”%” sign, followed by the program number (1 ... 9999).

When a program is loaded via the monitor interface, this program number is automatically created from the file name and inserted at the start of the program (123.DIN -> %123).

Every program can be directly started or called as a subprogram by other programs. Note that, in individual cases, the relevant parameters must be provided.

No distinction is made between main programs and subprograms by means of the program number.

Generally, program numbers ≥ 8000 are considered separately. Normally, cycle programs are stored in this range.

Block

A block consists of a statement, optionally preceded by a block number, (see below). A statement consists of the following elements:

- ”G” preparatory function
- ”M”, ”S”, ”T” auxiliary functions
- ”Q” fast outputs
- Parameter assignments
- Computer statement
- Comment

All elements must only be programmed once in a block. Exceptions are ”Fast outputs” and ”Parameter assignment”. These may be specified more than once in a block.

Computer statements are preceded by a colon. They must occur alone in a block and may only be combined with comments. A block may have a maximum length of 256 characters.

Program loop and branch

For program loops or branches, a statement can be preceded by block numbers. These must be assigned within the program in ascending order.

Program end

The program end is indicated by an M function or a G function:

- M30 = main program end
- G99 = subprogram end

Comments

Comments are limited by round or curly brackets and read over at runtime.

A comment can occur at the end of the block or alone in a block. The line length of 256 characters must also not be exceeded with a comment. If a comment is opened with a bracket, the line end is automatically seen as the comment end.

When using the curly brackets for comments, it is possible to suppress them during importation via the monitor interface into the control. This is defined by the machine constant MK_NCPROG_OHNE_KOMMENTARE. The comments are then not saved in the control.

When using computer statements, comments must generally be placed in curly brackets.

Example:

G17 (plane selection XY plane)
{This is also a comment}

G preparatory function

The preparatory function G is either only valid in its block (non-modal function) or replaces another function which was valid previously (modal function).

Modal G functions are combined in groups, in which only one function is valid at a time.

Modal functions

Modal functions remain valid until they are replaced by a new modal function of the same group. These functions are divided into the following groups:

Group	Function
a	Interpolation type
b	Change-over between path and route operation
c	Plane selection
d	Dimensions
e	Tool correction
f	Coordinate shift
g	Speed agreement
h	Functions, which do not mutually deactivate each other, but rather which are deactivated by themselves or a special deactivation function

When a subprogram is called, the valid modal states of the function groups a ... d are saved. In the subprogram it is then possible to activate new modal functions. The subprogram return then sets the modal functions to the saved state. This applies to all eight possible subprogram levels.

An exception is a program termination, which is either triggered by pressing the **<STOP>** key or by an error. During this process, the control is switched back to the path control operation and into the reference dimension system.

These and additional modal functions can be defined at the end of the program or after a program termination with the aid of the machine constant MK_VOREINSTELLUNG.

Block preprocessing and time synchronization

When a CNC program is processed, a distinction is made between program interpretation and program execution. Generally, a DIN block is not executed at the same time as its interpretation. Rather, for many functions it is important that the blocks are interpreted in advance. For example, the ”Look Ahead” function (G60 X0) determines an optimum speed profile via the interpreted interpolation blocks, which is only possible if a sufficient quantity of blocks are interpreted in advance.

Program execution and interpretation take place, in principle, in parallel and are only coupled via a prebuffer (FIFO memory), into which interpolation and other orders for execution are entered. This decoupling is first required before the program can be processed quickly. Therefore, the maximum number of blocks in this FIFO plays an important role on the speed at which a NC program can be processed. This number can be set using the machine constant MK_LAH_VORLAUFTIEFE to up to 1024 blocks. However, for most applications 16 blocks is more than adequate. For more demanding applications 256 or more blocks may also be useful.

It may be necessary at critical program points to stop this block preprocessing in order to synchronize the interpretation with the execution of the preceding blocks. Such a time synchronization is carried out automatically if an access is made to a time-synchronized parameter in the parameter field or a time-synchronized G function is programmed.

Even when ”Look Ahead” is activated, a time synchronization always means that the axes on the path come to a short-term stillstand as a consequence, because no subsequent speed can be determined without a successor block.

In the table containing the G functions, the time-synchronized G functions are indicated by the index ”s” (📖 91).

In addition to these functions there are also commands, which do not have an executing function, but rather merely affect the sequential program flow or only affect the interpretation of the program in another way. These functions are already completely executed during the interpretation, but are not placed in the prebuffer and, therefore, do not result in the path motion being interrupted.

These commands include:

- Parameter field assignments (P2000=100)
- Formula (P2001=P2000+50)
- Some G functions (e.g. G20, G22)

All the G functions concerned are indicated in the table containing the G functions by an index (📖 91).

3.2 G functions

Syntax

A DIN block with a G function always has the following structure: the letter ”G” is followed by the number of the G function. This is then followed by the parameters, which are each formed by their address letters and the corresponding value.

G(number) [address identification(address value)] ...

Possible address identifications are

- **X ,Y ,Z ,A ,B, C, U, V, W, x, y, z, a, b, c, u, v, w** as designations for axes, referred to as AXES in the following.
- **D, F, H, I, J, K, L, R** as G function-specific flow point parameters
- **E, O** as G function-specific integral parameters

As address values, any flow point numbers alternatively with exponent (e.g. $-23.475e^{-3}$) are valid for axis addresses and flow point parameters. For all the others, integers in the range of -32767 to +32767 are allowed.

As an alternative to direct input, all the above-mentioned addresses can also be programmed indirectly. This is displayed by the specification of ”P”. The programming takes place via the P field. Instead of the numerical value, the index on the P field is then assigned to the address letter.

Example

G0 X=P1000	For X, the value contained in the P field in the index 1000 is then valid.
G0 X=PP1100	For X, the value, whose index is contained P field in the index 1100 is valid (indicated twice).

A DIN block does not necessarily have to contain a G function. Some of the functions have a modal effect and do not need to be repeated. The above specifications are then valid for the rest of the block.

3.2.1 Overview of G functions

i Function is a pure interpreter function (📖 90)

s Function is time synchronized (📖 90)

1) Function is deactivated at the end of the program or in the event of a program termination and with modal branches

2) Function is deactivated at the end of the program or in the event of a program termination

3) Function is deactivated in the event of a program termination

4) Function is deactivated in the event of a program termination and with modal branches

No.	Meaning	Group	Attribute
00	Point-to-point positioning, high rate	a	
01	Linear interpolation	a	
02	Circular interpolation, clockwise	a	
03	Circular interpolation, counterclockwise	a	
04	Dwell time		
05	Spline interpolation via the programmed grid points	h1)	i
06	Polynomial interpolation with coefficient programming	h1)	i
10	Definition of an error recovery item after an error		

No.	Meaning	Group	Attribute
16	Definition of the principal axes of the current NC channel	h	s
17	Plane selection (XY)	c	i
18	Plane selection (XZ)	c	i
19	Plane selection (YX)	c	i
20	Block jump, optionally with condition check		I
21	Synchronous parameter field assignment		
22	Subprogram call, optionally with condition check and start of a new NC channel		I
24	Define negative traversing range limit	h	
25	Define positive traversing range limit	h	
26	Activate/deactivate traversing range limits	h	
27	Block jump with repetition counter		i
30	Route operation	b	s
31	Path operation	b	s
33	Coupling between path and rotation axes ”on”	h[2)]	
34	Path coupling ”off”	h	
36	Modal oscillation ”on”	h[3)]	s
37	Modal oscillation ”off”	h	s
40	Correction module ”off”	e	i
41	Correction module / left ”on”	e[4)]	i
42	Correction module / right ”off”	e[4)]	i
53	Deactivate temporary coordinate shift	f	i
54	Temporary zero shift of the current tool coordinate system	f[2)]	i
60	Exact positioning on/off	h[2)]	i
61	Stop block preprocessing		s
74	Home position approach		s
75	Change scaling factor for input units	h[2)]	i
76	Change scaling factor for pulse evaluation	h[2)]	
88	Basic rotation	h	s
89	Profile rotation	h[2)]	s
90	Absolute dimensions (reference dimension)	d	i
91	Incremental dimensions	d	i
92	Relative zero shift of the current coordinate system	f	i
93	Absolute zero shift of the current coordinate system		
96	Programming of spindle circumferential speed	g	
97	Programming of spindle speed		
99	Subprogram return		i
100	Polar coordinates: linear interpolation, high rate	a	
101	Polar coordinates: linear interpolation	a	
102	Polar coordinates: circular interpolation, clockwise	a	
103	Polar coordinates: circular interpolation, counterclockwise	a	
110	Polar coordinates: accept center		
110	Polar coordinates: accept center		i
112	Tangential correction ”on”	h	i
113	Tangential correction ”off”	h	i
114	6 axes transformation (X,Y,Z,A,B,C)	h[2)]	s
115	Sheath transformation	h[2)]	s
116	Rotation axis transformation (B,C) -> (A,B‘)	h[2)]	s

No.	Meaning	Group	Attribute
117	Reserved		
118	Reserved		
120	Coordinate shift via modal offset		s
121	Programming of the modal offset		s
122	Configuration of the effect of the traverse keys	h	
125	Non-modal comparative operation		i
130	Modal comparative operation	h[2)]	s
131	Export modal comparative operation	h	s
132	Inhibit modal program branching	h[2)]	s
133	Enable modal program branching	h	s
134	Non-modal waiting function, parameter field comparison		i
140	Noncircular grinding ”off”	E	s
141	Noncircular grinding with neg. direction of rotation ”on”	e[3)]	s
142	Noncircular grinding with pos. direction of rotation ”on”	e[3)]	s
143	Parameter setting of noncircular grinding phases		i
144	Configuration of a correction table for noncircular grinding	h	s
145	Definition of the correction values in the correction table	h	i
150	Modal program branching to an external event	h[2)]	s
151	Program branching to an external event		s
152	Non-modal waiting function, Q-Bit comparison		i
153	Wait for the scheduling of a NC channel		
158	Intermittent operation ”on”	h[4)]	
159	Intermittent operation ”off”	h	
161	Actual position acceptance		s
162	Define axis group		
175	Axis replacement	h	s
180	Modal travel ”on”	h[3)]	s
181	Modal travel ”off”	h	s
187	Handwheel coupling ”on”/”off”	h[3)]	
193	Absolute zero point determination in the current coordinate system	f	i
194	Basic offset shift	f	s
195	Shift all tool coordinate systems except for S0	f	
200	Geometry filter ”on”/”off”	h	i
201	Change the acceleration and deceleration ramps	h[2)]	i
209	Set the geometry counter		i
211	Transformation for two-axis articulated robot kinematics	h	s
222	Activate customer-specific output signal handler	h	
226	Reconfigure effect of the hardware limit switch	h	
231	Separation control system in the axis computer	h	s
232	Reserved for online cutter correction	h	s
233	3D axis correction	h	s
234	Direction-dependent height correction	h	s
250	Recording of axis positions or speeds	h	
251	Accept step response of an axis		s
252	Value input via the display device		s
253	Text display function, optional with error handling		s

3.2.2 G functions individual descriptions

Every G function is described in detail in the following. The functions are sorted numerically in ascending order.

Note!

For the preparatory functions, the code letters for the individual axes are explicitly specified. These refer to a configuration with a maximum of four axes. Other axis letters can also be used in another configuration. These must then be used accordingly in the preparatory functions. The used axes with the corresponding code letters are defined in the machine constants.

3.2.2.1 G00 Point-to-point positioning, high rate

The preparatory function G00 is used to program a high rate point-to-point positioning.

In contrast to DIN66025, route operation or path operation can be used. This is preselected by G30 / G31.

Syntax

G0 AXES R D E L

Meaning of the addresses

AXES	Target point coordinates of the axes
R	Radius, with which the following linear interpolation (G0, G1) should be connected.
D	Max. path deviation in the target point for grinding corners with the following linear interpolation.
F	High rate speed on the path (G31) or every axis (G30).
E, L	Selection of feed speed via speed (E) and increment (L). F=E*L.

Explanation

G00 can be programmed in path operation (G31) and in route operation (G30).

Path operation: All axes reach the programmed target point at the same time. Two directly successive G0/G1 functions can be connected with a tangential arc with the radius R. For this, parameter ”R” must be specified in the first block. The programmed radius is corrected to smaller radii by the control if the distance between the line-circle intersection point and the peak exceeds the length of one of the two lines or an R is also programmed in the following block. In the latter case, the available path length for the circle adjustment is shortened by half.

As an alternative to R, D can be used to program the maximum path deviation in the target point in order to define the grinding of the corner which is created between two G0/G1 blocks. The two blocks must also be programmed in direct succession here, otherwise the D is ignored. In contrast to the corner cut with R, no plane must be selected during the grinding of corners, since a polynomial (G6), and not a circle, is inserted between the two blocks. The interpolation takes place in the space.

The high rate speed programmed for F has a modal effect for G0. It does not have any affect on the speed for G1, G2 or G3. The value from MK_VBAHNMAX is preset.

The values programmed for E and L have a modal effect and do not have any affect on the high rate speed, but rather only on G1, G2 and G3. However, the resulting feed speed is only considered if a value not equal to zero was programmed for L. The value programmed for E is also stored time synchronously in P561 and can be used e.g. as the spindle speed.

The speed is limited in such a way that neither the max. path speed (MK_VBAHNMAX) nor the max. axis speeds (MK_VMAX) are exceeded. The acceleration and deceleration ramps on the path are limited in such a way that the max. ramps (MK_BESCHL, MK_BREMS) of the participating axes are not exceeded.

Route operation: All axes reach their target point independently from each other. The following block is only executed when all axes have reached their target point.

The high rate speed programmed for F has a modal effect for G0 and all programmed axes. It does not have any affect on the speed for G1, G2 or G3. The values from MK_VMAX are preset.

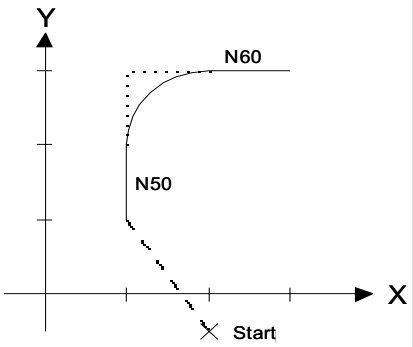
The speeds of the participating axes are limited to the respective max. axis speeds (MK_VMAX) in the machine constants. The maximum values (MK_BESCHL, MK_BREMS) configured in the machine constants are used as the acceleration and deceleration ramps, if smaller values were not selected with G201.

General: The preparatory function G00 has modal effect.

The end position of the axes can be programmed either in the reference dimension system (G90) or in the incremental dimension system (G91), under the addresses of the positioning axes.

The unit F is dependent on the axes which are programmed. If only linear axes are involved, input units/min (default: mm/min) is programmed in F. If only rotation axes are involved, 1/min is programmed in F. If both are involved, the unit F is dependent on whether the guide axes are linear or rotation axes (see G16).

Example



N10 G90	Change over to the reference dimension system and traverse the X axis to the position +20 mm and the Z axis to the position +200 mm.
N20 G0 X20 Z200	
N50 G0 X10 Y10	Select modal G function and approach the starting position.
N60 Y30 R10	The blocks N60 and N70 are connected with a tangential arc with a radius of 10 mm instead of with a 90° angle.
N70 X30	

3.2.2.2 G01 Linear interpolation

In the case of linear interpolation, the tool moves between the starting point, i.e. the current actual position, and the programmed end point (desired position) on a line. The programmed end point is reached by all axes at the same time.

Syntax

G1 AXES R D F E L

Meaning of the addresses

AXES	Target point coordinates of the axes.
R	Radius, with which the following linear interpolation (G0, G1) should be connected.
D	Max. path deviation in the target point for grinding corners with the following linear interpolation.
F	Feed speed on the path (G31) or every axis (G30).
E,L	Selection of feed speed via speed (E) and increment (L). F=E*L.

Explanation

G01 can be programmed in path operation (G31) and in route operation (G30).

Path operation: All axes reach the programmed target point at the same time.

Two directly successive G0/G1 functions can be connected by a tangential arc with the radius R. For this, parameter R must be specified in the first block. The programmed radius is corrected to smaller radii by the control system if the distance between the line-circle intersection point and the peak exceeds the length of one of the two lines or an R is also programmed in the following block. In the latter case, the available path length for the circle adjustment is shortened by half.

As an alternative to R, D can be used to program the maximum path deviation in the target point in order to define the grinding of the corner created between two G0/G1 blocks. The two blocks must also be programmed in direct succession here, otherwise the D is ignored. In contrast to the corner cut with R, no plane must be selected during the grinding of corners, since a polynomial (G6), and not a circle, is inserted between the two blocks. The interpolation takes place in the space.

The feed speed programmed for F has a modal effect. The value 0 is preset.

The values programmed for E and L have a modal effect. However, the resulting feed speed is only considered if a value not equal to zero was programmed for L. The value programmed for E is also stored time synchronously in P561 and can be used e.g. as the spindle speed.

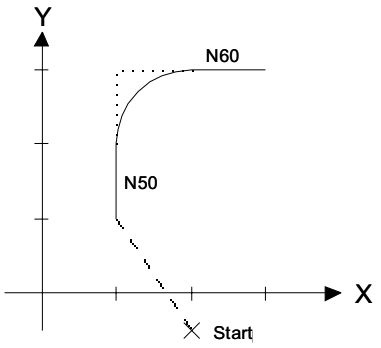
The speed is limited in such a way that neither the max. path speed (MK_VBAHNMAX) nor the max. axis speeds (MK_VMAX) are exceeded. The acceleration and deceleration ramps on the path are limited in such a way that the max. ramps (MK_BESCHL, MK_BREMS) of the participating axes are not exceeded.

Route operation: All axes travel with the programmed speed and reach their target point independently of each other according to the traverse route and the speed. The following block is only executed when all axes have reached their target point. The feed speed programmed for F has a modal effect for all programmed axes. The value 0 is preset for all axes. The speeds of the participating axes are limited to the respective max. axis speeds (MK_VMAX) in the machine constants. The maximum values configured in the machine constants (MK_BESCHL, MK_BREMS) are used as the acceleration and deceleration ramps, if smaller values were not selected with G201.

General: The preparatory function G01 has a modal effect.

The end position of the axes can be programmed either in the reference dimension system (G90) or in the incremental dimension system (G91), under the addresses of the positioning axes. The unit F is dependent on which axes are programmed. If only linear axes are involved, input units/min (default: mm/min) is programmed in F. If only rotation axes are involved, 1/min is programmed in F. If both are involved, the unit F is dependent on whether the guide axes are linear or rotation axes (see G16).

Example



N10 G31	Path operation ”on”
N20 G1 X0 Z10 C0 F2	Position X axis to 0 mm and Z axis to +10 mm with 2 mm/min, the C axis is also positioned to 0° at the same time.
N30 G1 C360 F60	C axis moves to position 360 degrees with the speed 60 revolutions per minute.
N50 G1 X10 Y10	Select modal G function and approach the starting position.
N60 Y30 R10 N70 X30	The blocks N60 and N70 are connected with a tangential arc with a radius of 10 mm, instead of with a 90° angle.

3.2.2.3 G02 Circular interpolation, clockwise, G03 Circular interpolation, counterclockwise

Definition of a circle or segment of a circle, clockwise or counterclockwise, with linear positioning of a linear axis (helix) or expanding radius (spiral) with additional positioning of a linear axis (conical helix).

Note!

When correction modules are used (e.g. TRC) only the normal arc can be programmed. The additional functions are not allowed!

Syntax

G02 AXES D I J K R F E L

G03 AXES D I J K R F E L

Meaning of the addresses

AXES	Target point coordinates of the three linear principal axes (default X,Y,Z), as well as start or target point coordinates of the three rotative secondary axes (default A,B,C).
D	Radius change on reaching the target point
I	Center coordinates of the first principal axis (X) or number of additional full circles
J	Center coordinates of the second principal axis (Y) or number of additional full circles
K	Center coordinates of the third principal axis (Z) or number of additional full circles
R	Interpolation radius
F	Path speed
E, L	Selection of feed speed via speed (E) and increment (L). F=E*L.

Explanation

With G02/G03, the interpolation on a circular path is programmed in the selected plane (G17, G18 or G19) (G02 clockwise, G03 counterclockwise).

Note!

It is always necessary to select a plane; there is no standard default setting!

To determine an arc, you need to specify three points. These are always the arc start point and arc end point, as well as either a center or a radius. This results in the following mandatory DIN address combinations for the determination of an arc, depending on the plane

G17: ”XY IJ” or ”XY R”
G18: ”ZX KI” or ”ZX R”
G19: ”YZ JK” or ”YZ R”

The target values can be specified both in absolute dimensions and in incremental dimensions.

If target values are missing, the corresponding circle start values are used. The specification of a center is always interpreted as a relative specification to the circle start point. Since there is always more than one circle center for the radius programming from a mathematical point of view, it is not possible to program a full circle in this circle determination mode. From the two possible centers, a negative radius selects the one which results in the larger arc. The combination of the DIN addresses of G2/G3, which are beyond the necessary specifications, results in the following geometries:

a) Helix with cylinder surface

The linear axis which is positioned vertically to the selected plane must be programmed. This specification results in a simultaneous linear interpolation of the axis between the circle start point and the circle target point. If an additional optional DIN address is not specified, then the helix covers a maximum of 360°.

G17: Z
G18: Y
G19: X

By specifying the number (n) of additional full circles, the helix can be extended to a maximum of (n+1)*360°.

G17: ZK
G18: YI
G19: XJ

b) Helix with cone surface

All the specifications under a) also apply here. However, a delta radius (**D**) must also be programmed. The cone radius then opens (D>0) or closes (D<0) evenly from the circle start point through to the circle target point.

G17: ZKD
G18: YID
G19: XJD

c) Spiral in the plane

All the specifications under b) are also needed here, however, without the linear positioning.

G17: KD
G18: ID
G19: JD

When the preparatory function G112 has been executed previously, the tangential correction of the rotation axis is possible, which is assigned to the selected plane (default: G17: C, G18: B, G19: A). If G112 was programmed with " Schlagzustellung der Rotationsachse " (Abrupt positioning of the rotation axis), the start angle of the rotation axis can be specified under the corresponding DIN addresses (A,B,C) for G2/G3. If the tangential correction is not switched on, the target angle of the rotation axis can be programmed. The rotation axis is then positioned linearly between the start and target angle on the circular path.

The principal and secondary axes which are assigned to the current NC channel can be selected with G16.

The path speed can be programmed under the address F. If the speed is not programmed, the speed that was last programmed is valid.

The path speed is limited in dependence on the radius and the allowed path acceleration according to the following formula:

$F_{max} = \sqrt{3,6 * R * B} * 1000$ [mm/min] with R=radius [mm] and B=acceleration [m/s^2]

The values programmed for E and L have a modal effect. However, the resulting feed speed is only considered if a value not equal to zero was programmed for L. The value programmed for E is also stored time synchronously in P561 and can be used e.g. as the spindle speed.

Special features in the circle calculation:

When a variable radius (D) is used, the control interprets the programmed speed as a specification for the larger radius. The angular velocity is constant over the complete G2/G3 profile, while the programmed path speed is only adjusted to the arc when the larger radius is reached. If a tangential correction is programmed, the path speed is adjusted to the accelerating performance of the corrected axis. The speed and acceleration values of the optional linear axis are not considered.

Any inaccuracies which may occur during the calculation of the circle geometries, which are in the window defined through the machine constants, are compensated by the control via the delta radius. In other words, all coordinate specifications for G2/G3 (target and center) are interpreted as a specification which does not need to be corrected.

Example

All the following examples for G2/G3 programming require that the XY plane (G17) has been switched on previously.

N10 G0 X50 Y100	Approach arc starting point
N20 G2 Y200 I0 J50 F200	Travels a semi-circle clockwise from X50 Y100 to X50 Y200 with a circle center of X50 Y150, a radius of 50 mm and a path speed of 200 mm/min.
N30 G3 X-50 Y100 R100	Travels a quarter circle in counterclockwise direction from X50 Y200 to X-50 Y100 with a radius of 100 mm.
G0 X0 Y0	Starting position X0 Y0
G3 X40 Y40 R100 D-60 K2	Travels a spiral in counterclockwise direction with a start radius of 100 mm, an end radius of 40 mm (R+D) and a covered angle of 810°.
G0 X0 Y0 Z0	Starting position X0 Y0 Z0
G3 X10 Y0 Z115 K12 R10	Travels a helix on a cylinder surface in counterclockwise direction with a radius of 10 mm, a length of 115 mm (Y) and a covered angle of 4500°.
G0 X0 Y-30 Z-115	Starting position X0 Y-300 Z-115
G3 X30 Y0 Z0 K10 R10 D20	Travels a helix on a cone surface in counterclockwise direction with a start radius of 10 mm, an end radius of 30 mm (R+D), a length of 115 mm (Y) and a covered angle of 3690°.

3.2.2.4 G04 Dwell time

The preparatory function G04 is used to program the dwell time. In other words, that the machine is motionless during the programmed time.

Syntax

G04 X

Meaning of the addresses

X: Dwell time in seconds

Explanation

The dwell time is programmed under the address letter ”X”, in steps of 0.01 s. The unit is 1 second.

The minimum time is dependent on the coarse interpolation cycle, the maximum time is 100 minutes (6000 s).

The modal preparatory functions which were valid before the dwell time remain effective. In a block with the preparatory function G04, an M function can still be programmed, which is processed after the dwell time.

Example

N30 G4 X5.1

Allow the dwell time of 5.1 s to pass, then continue with the following block.

3.2.2.5 G05 Spline interpolation

The spline interpolation is switched on with G05 and the specification of the involved axes and possibly additional parameters.

Syntax

G05 AXES I J K L

Meaning of the parameters

AXES	Validity identification (L=0) or starting position of the last block (L=1)
I	Boundary values (coupling to neighboring three-dimensional curves) 0: free boundaries (default) 1: start constant and end with free boundary 2: start free boundary, end constant 3: constant boundaries
K	Selection of the polynomial degree
	0: polynomial, third degree (second derivation not constant, default) 1: polynomial, fifth degree (second derivation constant)
J	Tangential correction of the rotation axis 0: relative to the starting position of the axis (default) 1: positioning of the rotation axis in the starting point of the spline curve on the tangent angle
L	Identification 0: open curve (default) 1: closed curve

Explanation

The grid points are specified by the following specification of the coordinates of the axes. The number of grid points is random. The end of a profile definition, which should be traveled through with the spline interpolation, is indicated by a G05 without parameters. The following positions are then approached in the programmed interpolation type. A spline, which only extends over one block, is executed as a normal linear block.

The axes involved in the spline interpolation are indicated by the specification of a validity identification for the respective axis letter. All the other axes, which are programmed in the following blocks, are interpolated linearly. They arrive at their target at the same time as the spline axes.

If a closed curve should be constant in the start and target point at the same time, the control must already know the starting point of the last block at the beginning of the profile. This must then be explicitly specified under the axis addresses for the G05 and the identification L set to 1.

In a spline sequence, the grid point derivatives are calculated with the aid of the neighboring grid points. On the left and right boundary of a spline sequence, the coupling to the neighboring profile segments can be affected via the address I. A difference must be made between free boundaries and the constant coupling, where a constant connection with circular sections can only take place in a principle plane.

In the case of a constant coupling at the end, the G05 must be followed by a linear or circular block. The effect on the rotation axis in the starting point of the spline curve with the address J applies only if the tangential correction of the axis was switched on before with G112 .

Notes for the use of spline interpolation:

The spline interpolation is also allowed in connection with a tool radius correction (TRC). It must be noted here that the TRC is executed before, i.e. the spline interpolation takes place via the corrected grid points.

A spline interpolation via grid points, which are programmed in the polar coordinate system, is possible.

The used algorithm for the spline interpolation also allows the backward interpolation of the blocks.

In a block with grid points, an M function and fast outputs can be programmed at the same time.

The use of modal comparative operations is possible. However, these must not be programmed within the spline sequence. In the event of a termination, the spline interpolation is switched off. The use of an interrupt program is possible. The programmed points are approached in the respective interpolation types. The use of the spline interpolation in the interrupt program is not allowed.

During the indirect programming of the grid points it must be noted that time-synchronized parameters cannot be accessed (e.g. current actual position).

Changing between absolute dimensions and incremental dimensions is possible. S/T change-overs and coordinate shifts are allowed. Jump functions (G20, G22, G125) are also allowed.

Directly successive blocks with identical grid points are ignored. If additional functions are programmed, these are also lost.

3.2.2.6 G06 Polynomial interpolation

Polynomial interpolation, third degree, in a block.

Syntax

G06 AXES[target;a_2;a_3] NAXS I J K E L F

Meaning of the parameters

AXES	Random axis address, whose polynomial coefficients (target, a2, a3) should be programmed.
NAXS	Secondary axis, which is interpolated linearly to the path movement (a2=0, a3=0)
I	Position of the segment in the polynomial line 0: intermediate segment (default) 1: the first segment 2: the last segment 3: one segment only
J	Tangential correction of the rotation axis 0 or not programmed - relative to the starting position of the axis 1: the rotation axis is first positioned in the starting point of this segment on the tangential angle
K	Type of curve - only programmed with the first segment of the curve 0: an open curve (default) 1: a closed curve
F	Path feed
E, L	Selection of feed speed via speed (E) and increment (L). F=E*L.

Explanation

With G6, a polynomial, third order, is programmed in the following form for every axis

$f(p) = a_3p^3 + a_2p^2 + a_1p + a_0$

Besides the target position of the respective axis, the coefficients a_2 and a_3 are also specified. The coefficient a_0 corresponds to the starting position of the axis and a_1 is calculated internally. p is the position on the chord, which results between start and target position of the involved axes.

Example

G06 X[100;0.1234;-0.00182] Y[15;0.00145;-0.0003067] A30 I3 F2000

3.2.2.7 G10 Definition of a restart position

G10 can be used to set restart positions in the program, at which the processing can be resumed after an error.

Syntax

G10 AXES

Meaning of the addresses

AXES	Axes, which should be repositioned for the restart.

Explanation

After a minor error has occurred, the ”Start nach G10” (Start after G10) command (see ”MC and NC Software Manual”) can be used to resume program processing again at the last programmed G10 prior to the error. For this, the G10 saves the following information:

- ► Absolute/relative dimensions (G90/G91)
- ► Path/route operation (G31/G30)
- ► Main plane (G17,G18,G19)
- ► Modal G function (G0,G1,G2,G3)
- ► Subprogram plane
- ► Tool and workpiece coordinate system (T,S)
- ► Actual positions of the programmed axes
- ► Status of the correction module (G40,G41,G42)

Please note that the G10 does not save either the state of the parameter field nor the zero points of the coordinate systems. The state of other NC functions like spline (G5) and radius transfer (G1 with R) are also not saved.

If possible, only use the G10 at the start of completed profile sections.

For the ”Wiederanlauf nach Fehler” (Restart after error) function, the machine constant MK_FEHLERRESTART must be set, otherwise a termination will also occur after a minor error and processing cannot be resumed at the point of interruption.

Example

N80 G10 X0 Y0	Block 80 is selected as the restart block and the current positions of the X and Y axis are saved as the restart position.

3.2.2.8 G16 Selection of the principal and secondary axes of the current NC channel

With G16, the principal and secondary axes of the current NC channel which are involved in the three main planes can be freely selected and thus the preset axes X,Y,Z and A,B,C can be replaced.

Syntax

G16 AXES

Meaning of the addresses

AXES	Axes, which should be selected as the principal and secondary axes. 1: first principal axis (X) 2: second principal axis (Y) 3: third principal axis (Z) 4: first secondary axis (C) 5: second secondary axis (A) 6: third secondary axis (B)

Explanation

The selection of other principal and secondary axes is necessary if different axes to the preset axes should be interpolated with G2/G3, corrected using the TRC or considered during Look Ahead.

If the first principal axis is a linear axis, then the second and third principal axis must also be a linear axis and all secondary axes must be rotation axes. All axes, which should be involved in the three main planes, must be programmed. The programming of the feed speed takes place with all interpolation tasks, in which one of the principal axes or another linear axis is involved, in mm/min.

If the first principal axis is a rotation axis, no other axes must be programmed for G16. The programmed axis becomes the master axis. The programming of the feed speed for all interpolations, in which this axis is involved, is made in 1/min.

G16 can be programmed without parameters in order to restore the default configuration with X,Y,Z and A,B,C.

Example

G16 U1 V2 W4	Selection of the U and V axis as linear principal axes and W as a rotative secondary axis, valid in the X-Y plane (G17).
G17	Select U-V plane.
G0 U0 V0	Approach starting point for circular interpolation.
G2 U100 R50	Interpolate semi-circle in the U-V plane.
G16	Restore default configuration.

3.2.2.9 G17, G18, G19 Plane selection

The preparatory functions G17 ... G19 are used to select the corresponding plane for various functions, e.g. circular interpolation in two axes.

Syntax

G17
G18
G19

Explanation

Selected plane:

G17	X-Y plane
G18	X-Z plane
G19	Y-Z plane

The selected plane is valid modally in the program. ”G17” is automatically valid after the end of the program.

A plane must be selected when a program is first started. A standard default setting is not provided.

Example

G18	The plane Z-X is selected.

3.2.2.10 G20 Block jump, optionally with condition check

Program branching in a program with programming of the jump target and an optional jump condition.

Syntax

G20 X E

Meaning of the addresses

X	Jump target
E	Index of the Q-Bit (direct programming) or of the parameter field (indirect programming) for the condition check.

Explanation

With G20, a jump is programmed within a program. If the DIN address E is also specified, the execution of the jump is made dependent on a condition.

If E is programmed directly, the function is time synchronized and the value of E is interpreted as an index of a Q-Bit. The jump is only executed if the Q-Bit is ”1”.

In the case of indirect programming, the jump is only executed if the contents of the specified parameter is not equal to ”0”. A time synchronization only takes place if it is a time synchronized parameter.

Example

N20 G20 X50	After block 20, block 50 is processed.
G20 X1 E14	Continue with block 1 if the Q-Bit 14 contains a value not equal to 0. Otherwise continue with the block after G20.

3.2.2.11 G22 Subprogram call, optionally with condition check and start of a new NC channel

Call of a subprogram as a separate program or within the current program, optionally dependent on a condition and programmable with repetition. The subprogram can also be started in a new NC channel.

Syntax

G22 AXES K L J I E

Meaning of the addresses

AXES	Axes, which should be transferred to the new NC channel (K).
K	Number of the new NC channel (> 1)
L	Number of the subprogram {1 ... 9999}
J	Block number
I	Loop counter
E	Index of the Q-Bit (direct programming) or of the parameter field (indirect programming) for the condition check.

Explanation

Function G22 allows a subprogram call, which can also be executed conditionally via the DIN address E on request and which can be executed in a separate NC channel via the DIN address K.

If E is directly programmed, the function is time synchronized and the value of E is interpreted as an index of a Q-Bit. The jump is only executed if the Q-Bit is ”1”. In the case of indirect programming, the jump is only executed when the contents of the specified parameter is not equal to ”0”. A time synchronization only takes place if it is a time synchronized parameter.

The program processing branches to the called subprogram L, the main program is continued after the return from the subprogram with the block following the subprogram call.

Up to seven subprogram calls can be nested.

Functions which have a modal effect are still valid after the return from a subprogram. In the subprogram, the state of the main program is initially valid, yet this can be changed.

By specifying a block number under the address J, it is possible to start with a specific block number of the subprogram. If no program number L is specified, the subprogram is called within the current program. The program processing then branches to the specified block number. The program is continued like with a normal subprogram call after G99 with the block following G22. It is important to note here that the block numbers must always be assigned numerically in ascending order!

It is also possible to program a number of repetitions under the address I. That means that the block following G22 is started when the program L was called I-times.

Starting a second NC channel

The subprogram can also be executed in parallel to the current program. For this, in K the number of the NC channel must be specified, in which the program should be started. The specified channel number must be greater than the number of the current channel (normally 0) and smaller than the maximum channel number in the machine constant MK_KANALANZAHL.

An NC channel can only activate the axes, which have been assigned to it. If axes should be traversed in channel 1, these must be transferred to the new channel at the start of the NC channel with G22. Channel 0 can retrieve the transferred axes with G153, as soon as channel 1 has been ended with M30.

The program in channel 0 can only be ended when the program in channel 1 has also been ended. The synchronization takes place automatically with M30.

Example

N120 G22 L1100 I4	Fourfold call of the subprogram %1100
N270 G22 J4000	Start of a subprogram in the same program file, which begins with block number N4000.
G22 L8000 J500	Call of the subprogram %8000 from block 500.
G22 L395 I3 E=P3012	Conditional call of the subprogram %395 up to 3-times in succession. The loop is ended prematurely if the parameter field has the value 0 at the position 3012.
G22 L9001 K1 U0 V0	Start of program %9001 in the second channel with transfer of the U and V axis to the new NC channel.

3.2.2.12 G24 Define positive traversing range limit

Definition of a traversing range limit in positive traversing direction.

Syntax

G24 AXES

Meaning of the addresses

AXES	Positive traversing range limit of the specified axes

Explanation

If axis addresses are specified with the preparatory function, the programmed values are entered into the parameter field from P304, related to the machine zero point defined by the basic offset. If, on the other hand, no other DIN address is specified except for the preparatory function, the parameter field is occupied again with values preset in the machine constants.

To activate the traversing range limits shifted with G24, a G26 must then be programmed with the corresponding axis letters.

Example

N100 G24	For all axes of the plant, the positive traversing range limits are predefined to the value defined by the machine constants.
G24 X2345.554	The positive traversing range limit of axis X is defined to the absolute position 2345.554.

3.2.2.13 G25 Define negative traversing range limit

Definition of a traversing range limit in negative traversing direction.

Syntax

G25 AXES

Meaning of the addresses

AXES	Negative traversing range limit of the specified axes

Explanation

If axis addresses are specified with the preparatory function, the programmed values are entered into the parameter field from P336, related to the machine zero point defined by the basic offset. If, on the other hand, no other DIN address is specified except for the preparatory function, the parameter field is occupied again with values preset in the machine constants.

To activate the traversing range limits shifted with G25, a G26 must then be programmed with the corresponding axis letters.

Example

N150 G25	For all axes of the plant, the negative traversing range limits are predefined to the value defined by the machine constants.
N200 G25 X-115	The negative traversing range limit of axis X is defined to the absolute position -115.

3.2.2.14 G26 Activate traversing range limits

Activation or deactivation of traversing range limits.

Syntax

G26 AXES

Meaning of the addresses

AXES	Validity identification (any value)

Explanation

If axis addresses are programmed in addition to the preparatory function, the values entered in the parameter field are transferred for the positive and negative traversing range limits and thus activated. If no other DIN addresses are programmed except for G26, the traversing range limits of all (!) axes are canceled.

Example

N250 G26 X0	The positive and the negative traversing range limit of the axis X is activated.

3.2.2.15 G27 Jump function with repetition counter

Programming of a loop, which should be passed n-times.

Syntax

G27 X Z

Meaning of the addresses

X	Jump target (block number)
Z	Repetition counter

Explanation

The programmed block number is carried out in accordance with the number in the repetition counter Z. The target block number must always be smaller than the block number, in which G27 is programmed, i.e. only returns are allowed. Failure to observe this leads to a termination with an error message.

The target block number must exist, otherwise an error message occurs during the program processing.

The maximum value which can be programmed under Z is 65535; if the value 0 is specified, the loop is passed through 65536-times.

Example

N10 G27 X10 Z3	The program executes the blocks between N10 and G27 three times before continuing with the block after G27.

3.2.2.16 G30, G31 Route or path control

The preparatory function G30 is used to switch the control to route operation and the preparatory function G31 is used to switch the control to path operation.

Syntax

G30
G31

Explanation

In path operation, all programmed axes travel a path, where the speed of the axes results from the programmed path speed. All axes reach their programmed end point at the same time.

In route operation, the programmed axes travel their distance at the same time, but independently from one other. All axes travel to the end point with the valid speed. The end point is not reached at the same time depending on the programmed distance.

The processing of the next block starts after the last axis has reached its target point.

The preparatory function G31 (Path operation) is preset after the control has been switched on or reset if another default setting has not been selected in the machine constants with MK_VOREINSTELLUNG.

Example

N5 G30	With block 5 the control is switched to the route operation.

3.2.2.17 G33 Coupling between path and rotation axes "on"

With G33, the coupling is switched on between the path or an axis involved in the path and a rotation axes.

Syntax

G33 A B C L E or
G33 I J K

Meaning of the addresses

ABC	Rotation axes, which should be corrected analog to the path. (Any value).
L	Path length, which should be covered during a complete revolution of the rotation axes.
E	Speed pre-selection of the rotation axes in 1/min. The resulting path speed is F=E*L.
I	Thread pitch in X direction. Coupling between X and A axis is active.
J	Thread pitch in Y direction. Coupling between Y and B axis is active.
K	Thread pitch in Z direction. Coupling between Z and C axis is active.

Explanation

G33 distinguishes between two variants using the programming.

The first variant is used to correct rotation axes to the feed on the distance (e.g. sewing technology). In this case, one revolution of the rotation axes is executed for each covered path. The cutting length is corrected for each block such so that a whole number of revolutions of the rotation axes is traveled.

The ramps and speeds on the path are corrected downwards if required such that the rotation axes are not overtaxed. With the second variant, the rotation axes is not corrected analog to the feed on the path, but to the feed of an individual axis. This variant is used in particular for thread grinding.

The programmed gradient is stored in P685 and the gradient correction is reset in P686. The total of P685 and P686 yields the effective thread pitch in mm/revolution. The gradient correction can be effected online by changing P686.

While the thread coupling is active, the involved rotation axes cannot be positioned independently. The coupling has a modal effect and must be switched off with G34. At the end of the program, the coupling of the first variant is switched off automatically. Both variants cannot be active at the same time. With the first variant,several rotation axes can be corrected analog to the path if these are programmed together for G33.

Example

G33 L5 A0	For each revolution of the A axis, 5 mm should be covered on the path.
G33 K1.5	The thread pitch should be 1.5 mm per revolution. Therefore, the C axis should make one revolution per 1.5 mm feed of the Z axis.

3.2.2.18 G34 Coupling between path and rotation axes ”off”

With G34, the path coupling, which was switched on with G33, is switched off again.

Syntax

G34

Explanation

G34 cancels the coupling between the path or axis involved in the path and the rotation axes again.

With the second variant of the G33, thread grinding, this is necessary to position the involved rotation axes alone, for example.

Example

G34	Coupling ”off”. Rotation axes is not carried anymore.

3.2.2.19 G36 Modal oscillation ”on”

Modal oscillation for an axis ”on”.

Syntax

G36 AXES I J K F

Meaning of the addresses

AXES	Axis to be oscillated and stroke in mm
I	Dwell time on the left stationary point (default 0)
J	Dwell time on the right stationary point (default 0)
K	Evaluation of I and J not programmed: no dwell time (I and J does not matter). 0: I, J specify the dwell time in seconds. 1: I, J specify the number of workpiece revolutions on the stationary point. Not implemented.
F	Oscillation speed [mm/min]

Explanation

The modal oscillating movement with programmable stroke and speed is switched on. The stroke refers to the position of the axis at the start of the block. A dwell time can be programmed at the stationary points. The times can be specified for both stationary points independently from each other. The movement is executed until it is stopped by G37. If more than one axis is programmed, the oscillation movement is executed for every axis independently. The specification of the dwell time and the oscillation speed then applies to every programmed axis.

Example

N10 G36 Z40 J1 F100	The Z axis executes an oscillating movement with a stroke of 40 mm and a speed of 100 mm/min. A dwell time of 1 sec takes place on the right stationary point.

3.2.2.20 G37 Modal oscillation ”off”

Syntax

G37 AXES

Meaning of the addresses

AXES	Validity identification for the axes, for which the modal oscillation should be switched off. Value = 1.

Explanation

With the preparatory function G37, the modal oscillation (see G36) is switched off. To achieve this, a 1 is programmed in a block with the preparatory function G37 under the address of the axis, whose modal oscillation should be switched off.

The axis is then stopped immediately, i.e. it does not travel its oscillation stroke to the end.

Example

N15 G37 Z1	In block 15, the modal oscillation of the Z axis is stopped.

3.2.2.21 G40 Deactivate tool correction

Switch off the tool path correction which was switched on by G41 / G42.

Syntax

G40 E

Meaning of the addresses

E	Selection of the departure strategy, only with standard TRC. 0: a departure block must follow G40 (default). 1: no departure block after G40, the axes remain on the last corrected position.

Explanation

G40 switches off the correction procedure which was switched on with G41 or G42 again. A tool radius compensation is provided by default.

The uncorrected target point is approached with the next traversing block. It must be noted that a traversing block must always be programmed for the TRC following G40, if E1 is not specified. The starting point of this block is on the compensated path, the target point of the block is uncompensated!

Example

N 100 G1 X10	Y10 Last profile block within the TRC
N 110 G40	TRC ”off”
N 120 G1 X0	Departure block

3.2.2.22 G41 Tool path correction - left, G42 Tool path correction - right

Switching on the tool radius compensation or an application-specific correction procedure for the tool path.

Syntax

G41 R J L
G42 R J L

Meaning of the addresses

R	Tool radius to be compensated
J	Tool orientation (value 0 ... 9)
L	Transfer type 0: linear, 1: circular

Explanation

The preparatory functions switch on the tool radius compensation or an application-specific correction procedure. The tool radius is transferred from the parameter field (P700), which is preset with a T change-over or can possibly be programmed directly with the code letter R. In the case of application-specific corrections, the dimensions are transferred via agreed interfaces (parameter field, machine constants). G41 activates a correction to the left, G42 to the right of the programmed path, related to the direction of the tool movement. The compensation applies to the selected plane (G17, G18, G19). After the compensation has been selected, the compensated path is approached with the following block. The starting point of this block is the current actual position, the target point lies on the compensated path (intersection point with following block).

Refer to the following table for the corresponding tool orientation J. It shows the relation between the coordinates of the cutting center (Xs,Ys) and the coordinates of the tool cutting point (Xp,Yp) for every orientation value.

J	Xp	Yp
0	Xs	Ys
1	Xs+R	Ys+R
2	Xs-R	Ys+R
3	Xs-R	Ys-R
4	Xs+R	Ys-R
5	Xs+R	Ys
6	Xs	Ys+R
7	Xs-R	Ys
8	Xs	Ys-R

The tool path correction works according to the intersection point procedure. If the profile transition angle is wide, the intersection point can be located very far from the path. In this case, linear intermediate blocks (L0, profile transition angle > 270 degrees) or a circular transition (L1, profile transfer angle > 180 degrees) are generated. The compensation is switched off by G40. Changing from G41 <-> G42 without intermediate G40 is allowed.

Example

N100 G41 R1	Tool radius correction with 1 mm tool radius ”on”.
N110 G1 X10	Approach block
N120 G1 X20 Y10	First profile block

3.2.2.23 G53 Deactivate temporary coordinate shift

Cancels the temporary coordinate shift in the current coordinate system again.

Syntax

G53

Explanation

The G53 function is sued to cancel a temporary coordinate shift carried out by G54. The originally defined zero points are valid again. This only applies to the current workpiece coordinate system.

Example

G53 S1	The previously made temporary zero shift in S1 is undone.

3.2.2.24 G54 Temporary zero shift

With the preparatory function G54, the zero point of the current workpiece coordinate system are temporarily shifted.

Syntax

G54 AXES]

Meaning of the addresses

AXES	Axes, whose zero point should be shifted.

Explanation

With G54, the zero point of the current workpiece coordinate system can be shifted. The shift takes place analog to G92, however, it only has a temporary effect. It is canceled at the end of the program or in the event of a program termination or by G53.

The special position of S0 must be noted. When S0 is shifted, all the other S coordinate systems are also shifted. If machine constant MK_S0T0_VERSATZ_ERLAUBT is not set, the shift in S0 is suppressed, however, it has an effect in all the other coordinates.

Example

G54 X10 Y10	The zero points of the axes X and Y in the current coordinate system (S) are each shifted by 10 mm.

3.2.2.25 G60 Exact positioning ”on”/”off”

Change over of the behavior of the path control at the end of the block.

Syntax

G60 X Y

Meaning of the addresses

X	Selector switch not programmed: exact positioning ”on”. 0: exact positioning ”off”, Look Ahead ”on”. >0: waiting time at the end of the block in seconds.
Y	Activation of exact positioning with tolerance margin monitoring. The programmed value does not have any meaning.

Explanation

The G60 function is used to effect the behavior of the path control at the end of a block of a traversing block. The basic setting is ”Genauhalt ohne Schleppabstandsüberwachung” (Exact positioning without tolerance margin monitoring).

Exact positioning without tolerance margin monitoring:

The target position of the block is approached via a deceleration ramp. The size of the ramp is defined in machine constants. The path speed at the end of the block is ”0”.

Exact positioning with waiting time:

The target position is approached analog to exact positioning. At this position, the time programmed under X has to elapse before the block is ended.

Exact positioning with tolerance margin monitoring:

After the target position has been reached, the control waits until the tolerance margin of the involved axes is within the configured exact positioning window (MK_SCHLEPPGENAUHALT). The maximum waiting time can be programmed for X, 10 seconds are preset.

Exact positioning ”off”:

The target position of the block is approached with the programmed speed if another traversing block follows. If the speed of the following block is lower, the speed is decelerated to this speed at the end of this block. The speed at the end of the block is effected by the profile course, i.e. the transition angle of the two blocks (”Look Ahead”).

The deceleration takes place via a ramp. If no other traversing block follows, the target position is approached analog to ”Genauhalt ohne Schleppabstandsüberwachung” (Exact positioning without tolerance margin monitoring).

Example

G60 X0	Exact positioning (with ”Look Ahead”) should now be switched off with immediate effect.

3.2.2.26 G61 Stop block preprocessing

Carry out time synchronization with coarse interpolator.

Syntax

G61

Explanation

With the G61 function, the block preprocessing of the interpreter can be temporarily stopped, i.e. the interpretation of the next block is stopped until the last block has been processed in the order buffer of the coarse interpolator.

If G61 is used within a profile, this forces the axes on the path to come to a temporary stillstand, since no connection speed can be determined without a connection block.

Example

N10 G1 X10 Z20	Block within the loop
...	Other blocks in the loop
N50 G61	Stop block preprocessing until the block before N50 has been executed.
N60 :P1999=P1999+1	Increase parameter for displaying the loops passed.
N70 G27 X10 Z33	Pass through loop 33-times.

3.2.2.27 G74 Home position approach

The preparatory function G74 is used to execute the home position approach.

Syntax

G74 AXES

Meaning of the addresses

AXES	Validity identification (any value)

Explanation

There are two home position approach possibilities.

If the function G74 is programmed without other addresses in a DIN block, the automatic home position approach is executed in the sequence which is defined in the machine constants.

If individual axes are specified, the home position approach is only carried out in the programmed axes.

The direction of the home position approach is defined by the corresponding machine constant of the axis, i.e. the direction corresponds to the automatic home position approach. An axis is programmed by the input of any single-digit number under the corresponding axis address.

Example

G74 X	The home position approach is carried out for the configured axes in the configured sequence.

3.2.2.28 G75 Scaling factor for input units

Setting the scaling factor of the individual axes for the programmed input units.

Syntax

G75 AXES

Meaning of the addresses

AXES	Specification of the axis and the factor

Explanation

G75 causes an extension or compression of the programmed profile before processing by internal correction modules. This can be non-symmetrical if different scaling factors are programmed for the individual axes. It must be noted here that only the respective programmed target coordinates, center coordinates and radius are scaled, i.e. circles remain unchanged as circles and are never expanded into ellipses.

If a negative scaling factor is used, it is mirrored about the respective axis. The programming of the function without the specification of axes sets all the factors to ”1” again.

Example

G75 X2 Y-3	Scaling factor 2 for X and -3 for Y
G1 X10 Y5	Travel to X20 and Y-15

3.2.2.29 G76 Scaling factors for pulse evaluation

Setting the scaling factor for the pulse evaluation of the individual axes.

Syntax

G76 AXES

Meaning of the addresses

AXES	Specification of the axis and the factor

Explanation

G76 can be used to change the pulse evaluation (pulses per distance) of the axes. This corresponds to a linear transformation, as made necessary e.g. by a change-over of the gearbox stage.

Like G75, G76 causes an extension or compression of the programmed profile. However, in contrast to G75, the speeds and accelerations are also effected. The max. allowed speeds and accelerations from the machine constants are adjusted for this accordingly. The change to the pulse evaluation means that the resulting profiles are different geometrically to those programmed in the case of non-symmetrical scaling factors, e.g. circles turn into ellipses and the speed on the path is not constant anymore.

If a negative scaling factor is used, it is mirrored about the respective axis. The programming of the function without the specification of axes sets all the factors to ”1” again.

Example

G76 C-1	Invert the movement direction of the C axis.

3.2.2.30 G88 Basic rotation

Rotation of the systems S1 - S31 in the XY plane.

Syntax

G88 C X Y

Meaning of the addresses

C	Basic rotation angle in degrees.
X, Y	Determination of the basic rotation angle from Delta X and Delta Y

Explanation

The basic rotation is used to define the position of a workpiece (S1 ... S31) relative to the machine workspace (S0). The rotation is effective for S1 ... S31 together about the same angle in the XY plane. In the reference workpiece coordinate system S0, the rotation has no effect, however, the rotation can be defined there, vertically in S0. It only becomes active when changing over to S1 ... S31.

When the rotation is activated, the actual positions of the X, Y and C axis are adjusted to the new position; a compensating movement does not take place.

The specification of the angle is an absolute specification.

The rotation is deactivated by programming the G88 without parameters or by the specification of angle 0°.

The function has a modal effect and remains active until the control is deactivated or reset.

Example

N1 G0 X10 Y15 S0 T0 N2 G193 X-10 Y50 S1 N3 G88 C30	With block 3, a rotation of the XY plane is activated for all systems S1 ... S31 about 30 degrees, relative to the unrotated S0. The rotation takes effect immediately, since it was activated in S1 vertically.

3.2.2.31 G89 Profile rotation

Rotation of the systems S1 ... S31 in the XYZ plane acts like the G88, however, is only effective within a program and in addition to G88.

Syntax

G89 A B C

Meaning of the addresses

a	Relative angle of rotation of the YZ plane in degrees
B	Relative angle of rotation of the ZX plane in degrees
C	Relative angle of rotation of the XY plane in degrees

Explanation

The profile rotation is used to define the position of a profile description on a workpiece. The rotation is effective for S1 ... S31 jointly. In the reference workpiece coordinate system S0, the rotation is not effective. However, the rotation can be defined there, vertically in S0. It only becomes active when changing over to S1 ... S31.

When the rotation is activated, the actual positions of the X, Y, Z, A, B and C axis are adjusted to the new plane location; a compensating movement does not take place.

The transformation sequence is always A->B->C.

The specifications of the angle are relative specifications.

The rotation is deactivated by omission of all the parameters and it is automatically deactivated at the end of the program or in the event of a program termination. During deactivation, the angle is reset to 0.

Example

N1 G0 X10 Y15 S0 T0	With block 4, a profile rotation of the YZ plane about 10° and the XY plane about 15° is activated for all systems S1 ... S31. The rotation is only becomes effective in block 5, since it was activated in S0 vertically.
N2 G193 X0 Y0 S1	
N3 S0	
N4 G89 A10 C15	The current actual positions are converted in block 5 to the rotated system.
N5 S1	

3.2.2.32 G90 Absolute dimensions (reference dimension)

Change-over to the reference dimension system.

Syntax

G90 AXIS R D

Meaning of the addresses

AXES	Validity identification of the axes (any value)
R	Validity identification for polar radius
D	Validity identification for polar angle

Explanation

In reference dimension programming, the programmed end point of the tool path always refers to the zero point of the selected coordinate system.

It is possible to change over individual axes or all the axes to reference dimension programming. If an axis is programmed, only this is changed over to incremental dimensions; the current dimensioning type remains valid for the others. If G90 is programmed without DIN addresses, all axes are changed over into the reference dimension system.

The preparatory function G90 has a modal effect and is only switched off by the preparatory function G91 (incremental dimension system).

Example

N3 G90	Change over all axes into the reference dimension system.

3.2.2.33 G91 Incremental dimensions (incremental dimension programming)

With the preparatory function G91, the reference dimension system changes over to the incremental dimension system. Generally, this can be done for all axes (without address parameters) and also for individual axes.

Syntax

G91 AXES R D

Meaning of the addresses

AXES	Validity identification of the axes (any value)
R	Validity identification for polar radius
D	Validity identification for polar angle

Explanation

In the incremental dimension system, the end point of the programmed axis is always programmed relative to its starting point. The sign specifies the direction of the axis movement; the starting point is the current axis position.

It is possible to change over individual axes or all the axes to incremental dimension programming. If an axis is programmed, only this is changed over to incremental dimension; the current dimensioning type remains valid for the others. If G91 is programmed without DIN addresses, all axes are changed over into the incremental dimension system.

The preparatory function G91 has a modal effect and is only switched off by the preparatory function G90 (reference dimension system).

Example

N7 G91	All axes changed over into the incremental dimension system.

3.2.2.34 G92 Relative zero shift

The preparatory function G92 can be used to effct an incrementally shift of the zero point of the current coordinate system.

Syntax

G92 AXES

Meaning of the addresses

AXES	Axes, whose zero point should be shifted.

Explanation

The incremental zero shift is programmed with the preparatory function G92 and the address letters of the axes, for which the zero point should be shifted.

Under the address letters of the axes, the value is programmed, by which the zero point of the axis should be shifted. With the preparatory function G92, the zero points of all desired axes can be shifted in a block.

If a T is programmed in the same block, the shift takes place in the current tool coordinate system (T0 ... T31), otherwise the current workpiece coordinate system (S0 ... S31) is shifted.

The special position of S0 must be noted. When S0 is shifted, all the other S coordinate systems are also shifted. If machine constant MK_S0T0_VERSATZ_ERLAUBT is not set, an error message is generated during the attempt to shift S0 and the program is terminated.

Example

N20 G92 X0.03 T1	The zero point of the X axis in T1 is shifted by +30 mm here.

3.2.2.35 G93 Absolute zero shift

G93 can be used to shift the zero point of the current coordinate system to an absolute programmed value.

Syntax

G93 AXES

Meaning of the addresses

AXES	Axes, whose zero point should be shifted

Explanation

The absolute zero shift is programmed with the preparatory function G93 and the address letters of the axes, for which the zero point should be shifted.

The new zero point offset of the coordinate system to be shifted is programmed directly under the address letters of the axes. The preparatory function G93 can be used to shift the zero points of all desired axes in a block.

If a T is programmed in the same block, the shift takes place in the current tool coordinate system (T0 ... T31), otherwise the current workpiece coordinate system (S0 ... S31) is shifted.

The special position of S0 must be noted. When S0 is shifted, all the other S coordinate systems are also shifted. If machine constant MK_S0T0_VERSATZ_ERLAUBT is not set, an error message is generated during the attempt to shift S0 and the program is terminated.

Example

N20 G93 Z50 T1	The zero point offset of the Z axis (tool length) is set in T1 to 50 mm here.

3.2.2.36 G99 Return

Subprogram end with return to the calling program.

Syntax

G99

Explanation

The preparatory function G99 must be contained in every subprogram as the last command.

The control continues the program in the main program with the block, which was programmed after the calling block.

No other address must be programmed in a block with the preparatory function G99.

Functions with a modal effect are only valid in the subprogram in which they were set. They are not valid anymore after the subprogram has been ended.

Example

N5 G99	Exit subprogram and continue processing in the calling program.

3.2.2.37 G100 Polar coordinates: linear interpolation, high rate

The preparatory function G100 corresponds in function to the preparatory function G00 during the programming of the target positions in the rectangular coordinate system.

Syntax

G100 AXES U W

Meaning of the addresses

AXES	Center coordinates
U	Polar radius
W	Polar angle

Explanation

The center of the polar coordinate system is defined by two addresses: X/Y for G17, X/Z for G18, Y/Z for G19. The assignment of the principal axes can be changed with G16. When the axes U and W are used as main axes, these cannot be used for the center programming.

The center coordinates can be programmed in absolute dimensions or in incremental dimensions.

The target position is defined by the polar radius U and the polar angle W in reference to the current center.

The radius and the angle can be programmed in absolute dimensions or in incremental dimensions.

The programmed center coordinates have a modal effect.

The preparatory function G100 has a modal effect.

Example

N100 G17 N110 G0 X50 Y0 N120 G100 X10 Y10 U20 W45	The zero point offset of the Z axis (tool length) is set in T1 to 50 mm here.

3.2.2.38 G101 Polar coordinates: linear interpolation

The preparatory function G101 corresponds in function to the preparatory function G01 during the programming of the target positions in the rectangular coordinate system.

Syntax

G101 AXES U W F E L

Meaning of the addresses

AXES	Center coordinates
U	Polar radius
W	Polar angle
F	Path speed
E, L	Selection of feed speed via speed (E) and increment (L). F=E*L.

Explanation

The center of the polar coordinate system is defined by two addresses: X/Y for G17, X/Z for G18, Y/Z for G19. The assignment of the principal axes can be changed with G16. When the axes U and W are used as main axes, these cannot be used for the center programming.

The center coordinates can be programmed in absolute dimensions or in incremental dimensions.

The target position is defined by the polar radius U and the polar angle W in reference to the current center.

The radius and the angle can be programmed in absolute dimensions or in incremental dimensions.

The programmed center coordinates have a modal effect.

The preparatory function G101 has a modal effect.

Example

```
N200 G18
N210 G0 X0 Z20
N220 G101 X10 Z10 U30 W45 F1000
```

3.2.2.39 G102/G103 Polar coordinates: circular interpolation

The preparatory function G102/G103 corresponds in function to the preparatory function G02/G03 during the programming of the circle with center coordinates in the rectangular coordinate system.

Syntax

G102 AXES W F E L
G103 AXES W F E L

Meaning of the addresses

AXES	Center coordinates
W	Polar angle
F	Path speed
E, L	Selection of feed speed via speed (E) and increment (L). F=E*L.

Explanation

The center of the polar coordinate system is defined by two addresses: X/Y for G17, X/Z for G18, Y/Z for G19. The assignment of the principal axes can be changed with G16. When the axes U and W are used as main axes, these cannot be used for the center programming.

The center coordinates can be programmed in absolute dimensions or in incremental dimensions. The center of the polar coordinate system also determines the center of the circle at the same time, therefore programming the circle radius is not necessary.

The target position is defined by the polar angle W in reference to the current center. The angle can be programmed in absolute dimensions or in incremental dimensions.

The programmed center coordinates have a modal effect.

The preparatory function G102/G103 has a modal effect.

Example

```
N100 G17
N110 G01 X50 Y0 F5000
N120 G103 X0 Y0 W180
N130 G102 X10 Y10 W0
```

3.2.2.40 G110 Polar coordinates: accept center

Transfer the reached set position as a new center in the polar coordinate system.

Syntax

G110 U W

Meaning of the addresses

U	Polar radius
W	Polar angle

Explanation

The current positions of the principal axes are transferred as new center coordinates and the target position programmed with the radius U and the angle W is approached with G101 from this new center.

If neither the radius U nor the angle W are programmed, no movement is executed.

The radius and the angle can be programmed in absolute dimensions or in incremental dimensions.

Example

```
N200 G17
N210 G0 X10 Y10
N220 G102 X20 W0 F1000
N230 G110 U20
N240 G102 W180
```

3.2.2.41 G111 Polar coordinates: center programming with angle and radius

Syntax

G111 U W

Meaning of the addresses

U	Polar radius
W	Polar angle

Explanation

The polar radius U and the polar angle W define the new center of the polar coordinate system. No axis movement takes place.

The radius and the angle can be programmed in absolute dimensions or in incremental dimensions.

Example

```
N10 G18
N20 G0 X0 Z50
N30 G101 X0 Z50 U50 W90 F2000
N40 G111 U50 W180
N50 G103 W90
```

3.2.2.42 G112 Tangential correction "on"

The tangential correction of the C axis is switched on.

Syntax

G112 AXES

Meaning of the addresses

AXES	Validity identification and mode for rotation axis to be corrected. Only one axis can be specified!

Explanation

For path profiles with tangential or approximate tangential transitions, an automatic tangential correction of a rotation axis is possible. For this, there are two modes, which must be programmed as a numerical value under the respective axis code letters.

Mode 1: Tangential correction from current position of the rotation axis (standard)

Mode 2: Tangential correction with abrupt positioning of the rotation axis at the start of the interpolation task on the angle programmed in the geometry function

The pure correction only has an effect on circle profiles; the abrupt positioning has an effect on all interpolation types.

Note!

This function does not calculate the current path vector in the case of linear interpolation, i.e. the rotation axis is not aligned automatically to the current path orientation.

The speed of the correction axis is corrected. The correction takes place up to the maximum permissible axis speed (MK). If this speed is exceeded, the path speed is reduced accordingly.

The function has a modal effect and can be deselected by G113.

Example

G112 C1	The tangential correction for the C axis is switched on in normal mode (pure correction without alignment on programmed angle).

3.2.2.43 G113 Tangential correction ”off”

Tangential correction ”off”

Syntax

G113 AXES

Meaning of the addresses

AXES	Axis, for which tangential correction should be switched off.

Explanation

The tangential correction switched on with G112 is switched off for the programmed axis. If none is programmed, the tangential correction is switched off for all axes.

Example

G113	The tangential correction is switched off.

3.2.2.44 G114 6-axes transformation

The 6-axes transformation is switched on or off to move the tool tip via 6 axes in the space at a constant speed.

Syntax

G114 I J K A B C E L

Meaning of the addresses

I	Tool length along the X axis at zero position of the A, B and C axis. This corresponds to the distance between the pivotal points of the C axis and the A axis. (Default 0).
J	Tool length along the Y axis at zero position of the A, B and C axis. This corresponds to the distance between the pivotal points of the A axis and the B axis. (Default 0).
K	Tool length along the Z axis at zero position of the A, B and C axis. This corresponds to the distance between the pivotal point of the B axis and the tool tip. (Default 0).
ABC	Identification for the rotation axes, which should be involved in the transformation. At least one is necessary. The value should be 0.
E	Identification: 0: tool path is programmed (default) 1: workpiece path is programmed
L	Identification: 0: kinematics with shifted pivotal points (default) 1: kinematics with common pivotal point of the A, B and C axis

Explanation

The 6-axes transformation allows for the consideration of different machine kinematics and tool geometries, e.g. spacial tool correction, tool orientation and interpolation of a tool path at a constant path speed. Both the path of the tool tip (tool path) and the path of the cartesian axes (tool path) can be programmed here.

The transformation is switched off through the programming of G114 without the specification of parameters and at the end of the program or in the event of a termination.

The sequence of the transformation is always C -> A -> B.

Example

G114 A0 J30 E1	Transformation ”on”, tool length along Y is 30 mm

3.2.2.45 G115 Convex surface transformation

Convex surface transformation ”on” or ”off”.

Syntax

G115 X Y Z A B C I

Meaning of the addresses

XYZ	Selection of lateral and longitudinal axis. 0: lateral axis ≠0: slope factor of the lateral axis (if programmed)
ABC	Selection of the involved rotation axis and radius at the start of the cone (when the position of the longitudinal axis=0)
I	without programmed axes: 0: switch off transformation temporarily 1: switch on transformation with old parameters again

Explanation

Sheath transformation is a process whereby cartesian coordinates are represented on a cone surface. The cone is defined via radius and slope. The cartesian coordinates are transformed during the axis output in coordinates for a rotation axis and a longitudinal axis. The rotation axis rotates the workpiece and the longitudinal axis moves the tool along the rotation axis. The third possible axis, vertically to the workpiece, is not transformed.

To switch the transformation off, G115 must be programmed without parameters.

When the transformation is switched on, the involved rotation axis can only still be moved via programming of the lateral axis.

Example

G115 X0 A50	Switch on the sheath transformation with X as the lateral axis, A as the rotation axis and any longitudinal axis. The radius of the cylinder, on whose surface the profile is represented, is 50 mm.

3.2.2.46 G116 Rotation axes transformation

Transforms a virtual axis system with the rotation axes B and C into a physical axis system with the rotation axes A and B‘.

Syntax

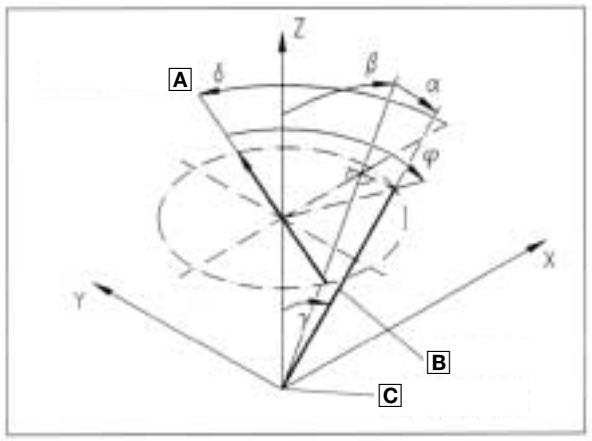
G116 C

Meaning of the addresses

c	Shift angle φ between path orientation and tool orientation.

Explanation

The function creates the virtual axes B and C and transforms their angles γ and δ with consideration of the shift angle φ into the angles α and β of the physical axes A and B‘. The following kinematic connection applies:



A Cutter direction
B Tool
C Tool tip

To switch on the transformation, the axes A and B must be configured. As long as the transformation is switched on, the physical axes A and B‘ are replaced by the virtual axes B and C and the programming of the corresponding physical axes is inhibited.

The domain of the angles α, β and γ is between –90° and +90°.

The transformation is switched off through the programming of G116 without parameters. At the end of the program or in the event of a program termination the transformation is automatically switched off.

Example

The following example should illustrate the use of G116 using the example of a cutting technology. For this, MK_KUNDE = ”SCHNEIDEN” (CUT) must be set. G41 switches on the tangential correction of the tool in this case. After the transformation has been switched on, the B axis is used to set the miter-box angle of the cutting tool.

...	Prologue
G60 X0	Look Ahead on
G17	Switch X/Y plane ”on”
G116 C90	Transformation on with shift angle φ = 90°
G41	Correction module for tangential correction of the C axis ”on”
G0 X100 Y50 B15 M14	Empty run to the start of the profile, swivel the tool by 15°
G3 X50 Y100 R50 F7000	First profile block
G1 X-50	Other profile blocks
G3 X-50 Y0 R50	
G1 X50	
G3 X100 Y50 R50 M15	Last profile block, M15 = empty running on
...	Even more profiles
G40	Correction module off
G116	Transformation off
...	Epilogue

3.2.2.47 G120 Absolute coordinate shift via modal offset

Setting the NC actual position to the programmed value by shifting the modal offset.

Syntax

G120 AXES

Meaning of the addresses

AXES	Axes, whose positions should be set.

Explanation

With G120, the modal offset is calculated so that the resulting actual position is equal to the programmed value for the axis address.

The effect is like a coordinate shift with G193, however on the modal offset. The modal offset can be reset with G121.

Example

N10 G90	In block 30, the internal actual position is set to 30 and the modal offset to 70.
N20 G0 X100	
N30 G120 X30	The displayed actual position (actual pos. + modal pos.) is not changed during this process.
N40 G0 X130	In block 40, 200=130+70 is traveled. The displayed actual position is 200.

3.2.2.48 G121 Programming the modal offset

Change of the modal offset with the transfer of the change amount into the NC actual position at the same time.

Syntax

G121 AXES

Meaning of the addresses

AXES	Validity identification and new modal offset

Explanation

When the traverse keys are used for traveling (modal or target point travel via PLC axes), the axes are positioned via the modal offset. This traverse movement can, in principle, be superimposed to the movement in the NC program, see G122. The desired position in reference to the current coordinate system of the control results from the total of NC actual position and the modal offset. The effect of the modal offset is like a coordinate shift.

G121 can be used to program the modal offset, in the most simple case, in order to zero it and to deactivate the shift triggered via the traverse keys. During this process, the modal offset is transferred into the NC actual position and the modal offset is reset.

However, the modal offset can also be used specifically to effect a coordinate shift asynchronously to the current program, e.g. in an interrupt program (see G130 and G150). Please note that an asynchronous shift via the tool and workpiece coordinate systems is not possible!

Example

G121 X0 Y0	The modal offset of the X and Y axis (P160,P161) is transferred into the NC actual position (P0,P1) and the modal offset is reset.
G121 X50 Y10	The modal offset of the X axis is set to 50 and the Y axis to 10, the NC actual position is reduced accordingly.

3.2.2.49 G122 Configuring the effect of the traverse keys

Syntax

G122 X

Meaning of the addresses

X	Traverse key mode

Explanation

G122 is used to configure the effect of the traverse keys to the axis movement. The value entered for X has the following meaning:

Mode 0: (Default) the traverse keys only have an effect in manual operation and not in the automatic program. When a program or an individual block is started, the modal offset is automatically reset, as described for G121.

Mode 1: In this mode, the traverse keys act as a modal offset to the programmed actual position. They act like a coordinate shift. The traverse movement takes place possibly as superimposed to a programmed movement and can take place both in manual operation and in automatic operation. The offset adjustment with the traverse keys can be transferred with G121 into the actual position.

Mode 2: In manual operation, the traverse keys have a normal effect. The resulting modal offset is automatically reset at the start of a program or an individual block. In automatic operation, the axes can be traversed with the traverse keys along a programmed movement. During this process, the programmed speed is replaced by the speed set with the traverse key. This also applies in the case of missing feed enable! If the speed is negative, the movement takes place backwards on the programmed profile.

In path operation, the traverse key of the first axis (axis 0) is used. In this case, both forward and backward movements are possible over the block limits. The number of blocks via which it is possible to move backwards is limited (MK_LAH_RUECKLAUFGRENZE). In route operation, the traverse keys of the axes programmed in the current block are effective. In this case, it is only possible to move forwards over the block limits and only if all axes have reached their target point.

Mode 3: This is a combination of mode 1 and 2. The traverse keys are effective both in manual operation and in automatic operation. The modal offset is not automatically reset. In the event that the feed enable is missing, it is possible to move forwards and backwards on the programmed profile like in mode 3.

Mode 4: The traverse keys are effective in manual operation and in automatic operation. However, the modal offset is automatically reset at the next opportunity. The forward and backward movement on the programmed profile is not possible.

After being switched on, the control has the state ”G122 X0”, provided another mode was not selected in MK_VOREINSTELLUNG.

Example

N33 G122 X1	Traverse keys act like a coordinate shift

3.2.2.50 G125 Non-modal comparative operation, parameter field comparison

Non-modal comparative operation without an effect on other, modal comparative operations.

Syntax

G125 X Z K I E

Meaning of the addresses

X	Operand 1
Z	Operand 2
K	Operator
I	Block number of the jump target
E	The P field index for storing the comparison result must be specified if I is not programmed

Explanation

G125 carries out a comparison K between the two operands X and Z. The comparative operations which are possible under K are:

0 : X = Z	X equals Z
1 : X > Z	X greater than Z
2 : X < Z	X smaller than Z
3 : $X \geq Z$	X greater than or equal to Z
4 : $X \leq Z$	X smaller than or equal to Z
5 : $X \neq Z$	X not equal to Z
6: $(X \wedge 2^Z) = 0$	Bit Z deleted in X
7: $(X \wedge 2^Z) \neq 0$	Bit Z set in X

If the programmed condition applies (the comparison result is ”true”), the program is continued at the indicated position (I). Otherwise the block following G125 is used. In contrast to G130, a ”true” result does not lead to the deletion of modal comparative operations.

If E is programmed, the comparison result (0 or 1) is also stored in the specified parameter.

Both constants and indexes of parameter fields are allowed as operands.

Example

G125 X=P701 Z100 K1 I20	If the value in P701 is greater than 100, the program processing is continued at block 20, otherwise with the block following G125.

3.2.2.51 G130 Modal comparative operation, parameter field comparison

Modal or non-modal comparative operation with an effect on other modal comparative operations

Syntax

G130 X Z K I E Y V J

Meaning of the addresses

X	First operand
Z	Second operand
K	Comparative operation
I	Target block number
E	Target program number
Y	Number of modal comparison (0 ... 6) or, if not programmed, non-modal comparison.
V	Index of an optional customer-specific comparative operation
J	Brakes 0: without ramp 1: with ramp (default)

Explanation

G130 carries out a comparison K between the two operands X and Z. Both constants and indexes of parameter fields are allowed as operands.

The comparative operations possible under K are

0 : $X = Z$	X equals Z
1 : $X > Z$	X greater than Z
2 : $X < Z$	X smaller than Z
3 : $X \geq Z$	X greater than or equal to Z
4 : $X \leq Z$	X smaller than or equal to Z
5 : $X \neq Z$	X not equal to Z
6: $(X \wedge 2^Z) = 0$	Bit Z deleted in X
7: $(X \wedge 2^Z) \neq 0$	Bit Z set in X

In the case of a ”true” result, all the on-going actions in the current NC channel are stopped. Two variants of continuing the program are then possible:

Program jump: The stopped actions are canceled and the program is continued with block I. In this case, the code letter E must not be programmed. The target block number I must be contained in the same program in which the comparison is also programmed. If the target block number I is not equal to the current block number N, a ”true” result leads to the deletion of all modal and non-modal comparative operations, which were made on the same or subordinate program planes.

Subprogram call: The stopped actions are saved, program E is placed in processing. In this case, the code letter E must be programmed; a target block number I must not be specified. If the subprogram is processed, the control continues processing at the interrupted position. A ”true” result does not lead to the export of modal or non-modal comparative operations.

It is the user's responsibility to ensure that all axes, which he uses in the subprogram, are returned to the positions which he found during the subprogram call.

Seven modal comparisons can be activated at the same time. With Y the index for the table line must be specified. A missing Y is interpreted as a non-modal comparison, i.e. the function is only interpreted once and does not remain effective in the background.

A programmed modal comparison remains active until

- another comparison was programmed at the same position of the comparison table (Y),
- the entry is exported with G131,
- this or another comparison (G130, G150, G151) becomes true on the same or a higher program plane and the result of the comparison is evaluated as a block jump,
- subprogram end; on the return to a higher program plane after the subprogram end, all comparisons are deactivated, which were activated in this subprogram,
- the comparison was completed (in the case of non-modal comparison),
- program end or program termination.

Note!

If several conditions are met at a specific time in the comparison table, which would result in a block jump, only the modal comparison with the lowest table index is executed.

The address "J" can be used to define whether the axes should be braked with or without ramp when the event occurs and when a traverse movement is interrupted as a result. The standard setting (J not programmed) is "Bremsen mit Rampe" (Brake with ramp).

Example

G130 X=P1 Z200 K2 Y1 I30	The comparison condition with the index 1 is entered. The jump to block 30 is executed if the contents of P1 become smaller than 200.
N20 G130 X=P1300 K5 Z1 I20	In block 20, wait until P1300 is not equal to 1.

3.2.2.52 G131 Delete modal comparative operation

Syntax

G131 Y

Meaning of the addresses

Y	Number of the modal comparison (0 ... 6)

Explanation

With the preparatory function G131, modal comparative operations, which were activated with G130, can be deleted again from the comparison operation table. For this, the number of the comparative operation must be programmed under the address Y.

If a number greater than 6 is programmed under the Y address or no number is specified, a program termination occurs with an error message.

Example

G131 Y1	The comparative operation set in the previous example with the index 1 is deleted again.

G131 Y1

The comparative operation set in the previous example with the index 1 is deleted again.

3.2.2.53 G132/G133 Inhibit/enable modal program branching

Syntax

G132
G133

Explanation

With G132, modal comparative operations, which were activated with G130 or G150, can be inhibited temporarily to protect critical program positions against an interruption.

As long as the modal comparative operations are inhibited, neither modal nor non-modal comparisons are executed with G130, G150 or G151. Modal comparisons are included in the table, but they are only executed after a G133. Non-modal comparisons are ignored.

Conditional program branches with G20, G22 or G125 are not affected.

The inhibit on the comparative operations is canceled again by G133, or through program end or program termination.

Example

G132	No interruption now
G0 X100 Y200 S0	Travel Z axis to tool change position
G0 Z0 M10	Get tool
G0 Z20	and lift it
G0 X0 Y0	Travel to start position
G133	Interruption allowed again

3.2.2.54 G134 Non-modal waiting function, parameter field comparison

Wait for a comparative operation to become true.

Syntax

G134 X Z K

Meaning of the addresses

X	First operand
Z	Second operand
K	Comparative operation

Explanation

G134 carries out a comparison K between the two operands X and Z. Both constants and indexes of parameter fields are allowed as operands.

In contrast to G130, no program branching takes place in the case of a positive comparison result. The G134 only blocks the program from being executed further in current channel until the comparison result becomes true.

The comparative operations possible under K are:

0 : X = Z	X equals Z
1 : X > Z	X greater than Z
2 : X < Z	X smaller than Z
3 : $X \geq Z$	X greater than or equal to Z
4 : $X \leq Z$	X smaller than or equal to Z
5 : $X \neq Z$	X not equal to Z
6: $(X \wedge 2^Z) = 0$	Bit Z deleted in X
7: $(X \wedge 2^Z) \neq 0$	Bit Z set in X

The function is not time synchronized and therefore it is also suitable for the fast responses in the range of maximum one interpolation cycle. No modal comparisons which were programmed previously with G130 or G150 are exported.

Example

G134 X=P16 K3 Z50	Waits for P16 ≥ to become 50.
...	Blocks, which should be executed afterwards

3.2.2.55 G140/G141/G142 Noncircular grinding ”on”/”off”

The functions G140 ... G142 are used to switch the ”Noncircular grinding” function on or off.

Syntax

G140
G141 X C D L K E
G142 X C D L K E

Meaning of the addresses

X	Total overmeasure in mm
C	Start area of the positioning as a position of the angle axis (with L=0 only). If it is not programmed, the start area of the positioning is defined by the starting position of the C axis
D	Grinding wheel diameter. If it is not programmed, the current diameter is taken from the tool data.
L	Profile 0: constant (default) 1: not constant
K	Noncircular grinding 0: without correction table (default) 1: with correction table (see G144, G145)
E	Noncircular grinding 0: from outside (default) 1: from inside

Explanation

G141 is used to switch on the ”Noncircular grinding” function, namely with a negative direction of rotation of the C axis during the profile creation. G142 activates noncircular grinding with a positive direction of rotation.

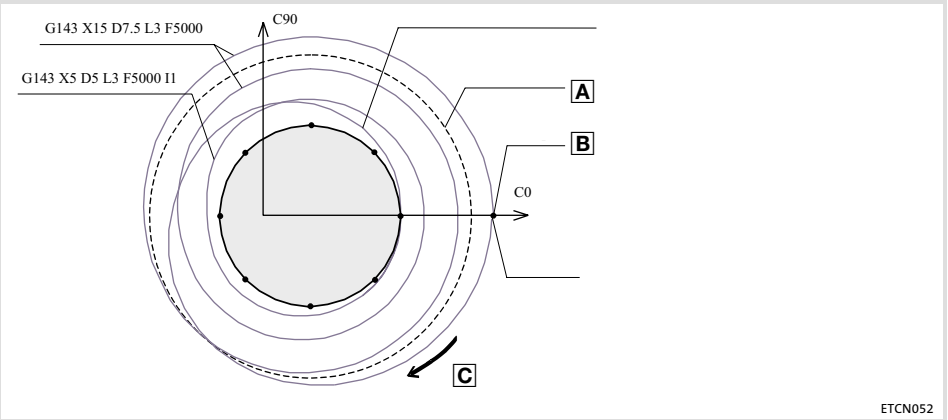
G140 switches the ”Noncircular grinding” function off again.

The individual grinding phases are called by G143, with the description of the finished profile as a subprogram, between G141/G142 and G140. The rotational direction of the profile description must match the rotational direction defined by G141 or G142.

Example

N10 G31	Path operation
N20 G01 X50 C0 F5000	Approach starting position
N30 G60 X0	Switch Look Ahead ”on”
N40 G142 D500 X20 L0 E0	Noncircular grinding for smooth profile ”on”
N41 G5 X24.142 Y-14.142 I3 L1	Switch spline interpolation for closed curve ”on”
N42 G143 X15 D7.5 F5000 L3 I0	Rough-grinding two revolutions
N43 G143 X5 D5 F5000 L3 I1	Finish grinding with a spark-out revolution
N44 G143 X-20 D20 F5000 L3	Withdraw
N45 G5	Switch spline interpolation off
N46 G140	Noncircular grinding ”off”
N50 G60	Switch Look Ahead ”off”
N60 M30	Program end
%3	
N10 X24.142 Y14.142	Subprogram for describing the finished profile
N20 X10 Y20	
N30 X-4.142 Y14.142	
N40 X-10 Y0	
N50 X-4.142 Y-14.142	
N60 X10 Y-20	
N70 X24.142 Y-14.142	
N80 X30 Y0	
N90 G99	

Position of X-axis in the Cartesian coordinate system:



A Start overmeasure
B Starting point
C Direction of rotation of the workpiece

The workpiece is only approached during the first revolution because the actual overmeasure is smaller than the defined overmeasure.

The withdrawal from the finished profile also takes place via G143, by programming the overmeasure and the positioning amount negatively.

3.2.2.56 G143 Parameters of grinding phases during noncircular grinding

With G143, the parameters must be programmed for every grinding phase, e.g. roughing, smoothing, fine finishing and sparking out.

Syntax

G143 X D C F L J I

Meaning of the addresses

X	Overmeasure for this phase in mm.
D	Positioning amount per revolution in mm.
C	Motion range of the C axis in which the positioning should take place. If it is not programmed, the positioning takes place via a complete revolution of the C axis.
F	Path speed during this phase in mm/min.
L	Program number of the subprogram in which the finished profile is described.
J	Block number in the current program, from which the finished profile is described.
I	Number of additional revolutions without positioning.

Explanation

G143 is needed to define the grinding phases during noncircular grinding. At least one grinding phase is necessary.

For L, the number of the program is specified, in which the finished profile is described. As an alternative to L, the block number, from which the finished profile is described in the current program, can also be specified with J. In any case, the description of the finished profile must be ended with G99.

The profile is described in a virtual X/Y plane, with the X position 0° and the Y position 90° corresponding to the C axis. For this, both cartesian coordinates (G1, G2, G3) and polar coordinates (G101, G102, G103) can be used.

When using the spline interpolation, G5 must not be programmed in the finished profile. Instead, it must already be switched on before the first G143 and switched off after the last G143.

The total number of workpiece revolutions of a grinding phase results from the programmed overmeasure and the positioning amount per revolution (X/D) plus the programmed number of revolutions without positioning.

Example

See G140/G141/G142

3.2.2.57 G144/G145 Programming a correction table during noncircular grinding

The correction table is used for the compensation of any systematic errors during the creation of a profile during noncircular grinding.

Syntax

G144 D
G145 C X

Meaning of the addresses

D	Distance of the grid points in the table in degrees
C	Value of the reference axis in degrees
X	Correction value in µm

Explanation

With G144, the correction table is configured and all correction values are initialized to 0. The grid points are defined by the following G145 commands. It is not necessary for all 720 possible grid points to be programmed here. The correction values of the not programmed grid points are initialized to 0.

The table contains the correction values for the X axis in reference to the modulo-360° position of the C axis. Up to 720 correction values can be stored in the table, i.e. the minimum distance between the grid points is 0.5°.

The correction takes place through the addition of the correction values on the axis setpoints of the X axis. A linear interpolation takes place between two neighboring grid points.

The uncorrected position values always appear in the display of the NC actual position.

The correction table must be defined before the noncircular grinding is switched on.

Example

G144 D10	Distance of the grid points should be 10°
G145 C330 X12	The correction takes place in the range of 320° to 20°, with the correction value at 320° and at 20° 0 µm. With 350° the greatest correction takes place with 13 µm.
G145 C340 X13	
G145 C350 X13	
G145 C360 X13	
G145 C10 X12	

3.2.2.58 G150 Modal comparative operation, Q-Bit comparison

Import or export of a modal comparison to an external event (Q-Bit).

Syntax

G150 X Y Z E V J L

Meaning of the addresses

X	Target block number
Y	Target program number
Z	State of the comparison
E	Index of the external event
V	Index of an optional customer-specific comparative operation
J	Brakes 0: without ramp 1: with ramp (default)
L	Parameter field index for saving the axis actual positions

Explanation

G150 carries out a comparison of the Q-Bit E (0 ... 63) on the state Z (0 or 1). This also applies for ”fast inputs”, which are also represented as Q-Bit.

In the case of a ”true” result, all the on-going actions in the current NC channel are stopped. Two variants of continuing the program are then possible with G150:

Program jump:

The stopped actions are canceled and the program is continued with the block X. In this case, the code letter Y must not be programmed. The target block number X must be contained in the same program in which the comparison is also programmed.

If the target block number X is not equal to the current block number N, a ”true” result leads to the export of all modal and non-modal comparative operations, which were made on the same or subordinate program planes.

Subprogram call:

The stopped actions are saved, program Y is placed in processing. In this case, the code letter Y must be programmed; a target block number X must not be specified. If the subprogram is processed, the control continues processing at the interrupted position. A ”true” result does not lead to the export of modal or non-modal comparative operations.

It is the user’s responsibility to ensure that all axes, which he uses in the subprogram, are returned to the positions which he found during the subprogram call.

The function still has a modal effect. It is deactivated if

- ► the entry is reset (see below)
- ► when the result of the comparison is positive and the resulting block jump
- ► at the end of the subprogram; in this case all comparisons are deactivated, which were activated in this subprogram.
- ► Occurrence of another modal event (G130, G150) in a higher program plane with a resulting block jump. Since in this case a return to the higher program plane must take place, all modal comparisons of the subordinate program planes are deactivated.
- ► Program end or program termination (stop or error)

Seven modal comparisons can be activated at the same time.

Note!

A modal comparison and a non-modal comparison must never be programmed simultaneously on the same Q-Bit with the same result. This may lead to non-defined results.

The address J can be used to define whether the axes should be braked with or without ramp when the event occurs and when a traverse movement is interrupted as a result. The standard setting (J not programmed) is ”Bremsen mit Rampe” (Brake with ramp).

If the address L is programmed, when the condition which was defined with E and Z applies, the actual positions of all configured axes are saved from the index in the parameter field specified for L. This function can be used to determine the position of the event independently of the configured brake ramp.

An entry is exported from the table of active Q-Bit comparisons, by programming the G150 with the specification of the Q-Bit number E without additional parameters.

Example

N100 G150 X200 Z1 E22	Branch to block number 200 if a 1 appears instead of 22 in the Q-field.
N200 G150 X500 E18 Z0 L1780	Branch to block 500 if Q-Bit 18 has the state 0; in addition the axes actual position are stored from P1780.
N300 G150 E22	Deactivate the modal monitoring of Q-Bit 22.

3.2.2.59 G151 Non-modal comparative operation, Q-Bit comparison

Execution of a non-modal comparison to an external event (Q-Bit).

Syntax

G151 X E Z V

Meaning of the addresses

X	Jump target block number
Z	State of the comparison
E	Index of the external event
V	Index of an optional customer-specific comparative operation
L	Parameter field index for saving the axis actual positions

Explanation

G151 carries out once (non-modal) a comparison of the Q-Bit E (0 ... 63) with the state Z (0 or 1).

In the case of a ”true” result, the program is continued with block X. The target block number X must be contained in the same program in which the comparison is also programmed.

If the target block number X is not equal to the current block number N, a ”true” result leads to the export of all modal comparative operations, which were made on the same or subordinate program planes.

The address J can be used to define whether the axes should be braked with or without ramp when the event occurs and when a traverse movement is interrupted as a result. The standard setting (J not programmed) is ”Bremsen mit Rampe” (Brake with ramp).

If the address L is programmed, when the condition which was defined with E and Z applies, the actual positions of all configured axes are saved from the index in the parameter field specified for L.

This function can be used to determine the position of the event independently of the configured brake ramp.

Example

N10 G151 X10 E31 Z1	The program does not continue with the program processing until the Q-Bit 31 has reached the state 0 (remains in position, as long as condition (Q[31] == 1) applies).

3.2.2.60 G152 Non-modal waiting function, Q-Bit comparison

Wait for a comparison to be become true on an externes event.

Syntax

G152 E Z

Meaning of the addresses

E	Index of the external event
Z	State of the comparison

Explanation

G152 carries out a comparison of the Q-Bit E (0 ... 63) with the state Z (0 or 1).

In contrast to G151, no block jump takes place in the case of a positive comparison result. G152 only blocks the program from being executed further in current channel until the comparison result becomes true.

The function is not time synchronized and therefore it is also suitable for fast responses in the range of maximum one interpolation cycle when fast inputs are used. No modal comparisons which were programmed before with G130 or G150 are exported.

Example

G152 E3 Z0	Waits for the state 0 of Q-Bit 3
G152 E3 Z1	Waits for the state 1 of Q-Bit 3
...	Blocks, which should be executed after a positive edge of Q-Bit 3

3.2.2.61 G153 Wait for the termination of a channel 1

G153 is used to synchronize the program end in channel 1 with channel 0.

Syntax

G153 K

Meaning of the addresses

K	Channel number of channel 1

Explanation

G153 is called in channel 0 in order to wait for the program end in channel 1 and in order to return the axes of channel 1 again to channel 0. The return of the axes is only possible through the termination of channel 1.

A channel can cannot be terminated from outside except for by a program termination, but rather only by a M30 at the main program level of the corresponding channel.

Example

G22 L10 K1 U0 V0	Start the program %10 in the second channel with transfer of U and V axis.
G1 X10 Y5 F100	The axes U and V cannot be traveled here.
G153 K1	Wait until the program is completed in channel 1.
G0 U0 V0	The axes U and V can be addressed again here.
M30	
%10	Program for the second channel
G0 U100 M23	
G0 V50 M25	
G0 U200 V150 M27	
M30	

3.2.2.62 G158/G159 Intermittent operation ”on”/”off”

The preparatory function G158 switches the intermittent operation on, during which the interpolation of the programmed profile is executed in strokes.

Syntax

G158 I J K
G159

Meaning of the addresses

K	Speed profile of every individual interpolation stroke 0: without intermit 1: discontinuous (default) 2: linear 3: sinusoidal 4: sinusoidal acceleration sine squared braking 5: sine squared
I	Feed duration in percent of the step length (default = 50)
J	Phase displacement in percent 0: feed on step end (default) 100: feed on step start

Explanation

During intermittent operation, all interpolation types on the profile are executed in strokes. The distance covered with every stroke on the path corresponds to the step length L for G1/G2/G3. With the combination of G158 and G33, it is possible with a sewing-machine (see sewing technology) to carry out the material transport in a limited angle range of a sewing motor revolution only.

Every stroke causes an acceleration and deceleration process of the material to be transported. The type of this acceleration curve is defined with the address K independent of the current path acceleration. The least load is placed on the mechanics in the case of sine-square form acceleration. However, double the speed is needed as for discontinuous acceleration to cover the same distance in the same time.

With I, the duration of the material transport is defined in relation to a full sewing motor revolution in percent. The value 50 corresponds to a half revolution. The smaller the programmed value, the greater the resulting values for maximum speed and acceleration.

The intermittent operation has an effect on profiles between M14 and M15/M16/M21. It also has a modal effect beyond the program end. It is switched off by G159.

Example

G158 K2	Activate intermittent operation with linear ramps
G0 X10 Y300 M14	Approach starting position, and select profile start (M14)
G0 A90	Position needle into the starting position 90° (0° = highest position)
G33 A0	Automatic coupling between path and rotation axis ”on”
G1 E500 L3	Select speed and stitch length
X300	Profile
Y10	
X10	
Y300	
M15	Select profile end
G34	Path coupling ”off”
G0 A0	Needle up
G159	Intermittent operation ”off”

3.2.2.63 G161 Accept actual position

Acceptance of the current actual positions of the axes as the new starting position for the next interpolation.

Syntax

G161 AXES

Meaning of the addresses

AXES	Validity identification of the axes, whose actual positions should be accepted.

Explanation

If it once becomes necessary in a DIN program to deliberately remove the controller enable of one or several axes temporarily, this function can be helpful. With G161, it is possible, after the controller enable has been restarted, to accept the possibly changed axis positions as a new starting position for the next traverse function into the interpreter.

If no axes are programmed, the actual positions of all axes of the current channel are accepted.

Example

M80	PLC temporarily removes controller enable and read enable
G161	Accept all axis actual positions

3.2.2.64 G162 Define axis group

Switches the forced coupling of two axes ”on” or ”off”.

Syntax

G162 LAX FAX

Meaning of the addresses

LAX	Code letter of the master axis, the programmed value must be 0
FAX	Code letter of the slave axis and coupling factor ≠0: Define axis group with the specified coupling factor 0: Delete axis group

Explanation

G162 is used to define an axis group. This function makes it possible to couple one or several slave axes by software via a user-definable factor to a master axis.

The corrected set position of the slave axis $\mathbf{P_F'}$ is calculated via the following formula: **PF’ = PF + m V PL - b**

$\mathbf{P_F}$= set position of the slave axis
$\mathbf{P_L}$= set position of the master axis
m = coupling factor for G162
b = start offset when switching G162 on

The start offset is calculated such that when the G162 is switched on it does not result in the slave axis moving: $\mathbf{b = m \cdot P_L}$.

If several slave axes should be coupled to a master axis or vice versa, G162 must be programmed once for every axis pair to be defined. The resulting total desired position including all the transformations (also G162) and coordinate system offsets (S0+Sn+Tn) is stored from P944 for all axes in the parameter field.

If no axis is programmed, all the axis couplings which were activated with G162 are deactivated.

Note!

The function does not limit the speed of the slave axis and the actual value difference between the master axis and the slave axis is not monitored. If necessary, a corresponding monitoring must be carried out in the PLC (see example).

If the feed enables of the slave axes exist, these axes can also be traversed when the coupling is switched on independent of the master axis. This can be helpful when setting up the machine, for example.

Example

The axes A and B should be coupled one to one to the movement of axis C after the home position approach. The starting positions and coordinate system offsets of all three axes should be equal when they are activated, i.e. b=0 and m=1.

G162	Delete axis group
G74 C0 A0 B0	Home position movement
G0 C0 A0 B0	approach all axes to be coupled to the same starting position
G162 C0 A1	Couple A axis to C axis
G162 C0 B1	Couple B axis to C axis
M174	Inform PLC that axis coupling has taken place

3.2.2.65 G175 Axis replacement

Replacement of two axes in terms of the input.

Syntax

G175 AXES

Meaning of the addresses

AXES	Validity identification of the two axis letters, which should be replaced; numerical value without meaning

Explanation

G175 can be used to program a replacement of the axis assignment. Two similar axes can be replaced by one another, i.e. only linear axes can be replaced by linear axes and rotation axes with rotation axes. All position specifications of an axis are referenced to the replaced axis after the replacement.

It is only possible to replace axes again after the original state has been restored again by programming G175 without further specifications.

Example

G175 U0 Z0	All path specifications for ”U” for the ”Z” axis are interpreted from here. On the machine, the Z axis traverses the paths programmed for U and vice versa.

3.2.2.66 G180 Modal travel ”on”

Traverse one or several axes modally with the programmed speed in the specified direction.

Syntax

G180 AXES F

Meaning of the addresses

AXES	Axes to be traveled. The programmed value specifies the traversing direction. > 0: positive traversing direction, < 0: negative traversing direction, = 0: only preselect speed.
F	Modal traversing speed.

Explanation

With G180, axes can be traversed modally in the specified direction, until the traversing function is deactivated again with G181. The further execution of the program is not blocked during this process, so that other axes can also be traversed in the program in parallel.

If several axes are programmed in a G180, the acceleration ramp of the individual axes is reduced such that all axes reach their programmed speed at the same time. If a traversing direction is specified and the axis is already modally traversed, the axis is accelerated to the new speed and direction without stopping.

If a feed speed F is programmed, this applies to all the axes programmed in the same block. However, if linear and rotation axes are involved, it only applies to the linear axes. Preselecting a speed for one or several axes, without starting them, is carried out by selecting the ”Verfahrrichtung 0” (traversing direction 0).

Modal travel axes cannot be positioned with geometry functions. They are stopped by G181, the stop button and by errors of class 3 and 4.

Example

N10 G180 X0 F1000	Preselect 1000 mm/min as the modal speed for X
N20 G180 A0 B0 F20	Preselect 20 U/min as the modal speed for A and B
N30 G180 X1 A1 B-1	Traverse X, A and B modally. X in positive direction with 1000 mm/min, A positive with 20 U/min and B negative with 20 U/min.
N40 G180 Y-1 F2000	Traverse Y with 2000 mm/min modally in negative direction
...	
N80 G181 X0 Y0	X and Y stop, A and B continue.
N90 G181	stop all axes started with G180

3.2.2.67 G181 Modal travel "off"

Stop one, several or all axes started with G180.

Syntax

G181 AXES

Meaning of the addresses

AXES	Axes to be stopped. The programmed value is without meaning.

Explanation

All programmed axes are stopped if they were started with G180. To stop all axes started with G180, it is sufficient to program this function without axes.

Example

see G180

3.2.2.68 G187 Handwheel

Switching the handwheel function for one or several axes on or off.

Syntax

G187 AXES

Meaning of the addresses

AXES	Evaluation factor of the handwheel for the assigned axis

Explanation

The value programmed under AXES is interpreted as an evaluation of the handwheel path for the selected axis. The handwheel function works superimposed to all the other traversing movements of the axes.

The resolution of the handwheel is configured with MK_IMPULSE and MK_WEG. The product from the evaluation factor and the value from MK_HANDRADFAKTOR acts as a multiplier on the path of the handwheel.

No handwheel is active by default.

The assignment of the axes to the configured handwheels is defined with MK_HANDRADZUORDNUNG. This is necessary if more than one handwheel exists in the machine.

The function is created internally via the traverse key codes 105 ... 109 by default.

Example

N30 G187 X0.2	The pulses of the handwheel are evaluated with the factor 0.2 added to the position of the X axis.
N40 G187 X0	Deactivate the assignment of the handwheel to the X axis.

3.2.2.69 G193 Set absolute zero point

The current position of the programmed axes is set to the specified position value in the current coordinate system.

Syntax

G193 AXES

Meaning of the addresses

AXES	Axes, whose coordinate zero point should be set.

Explanation

With G193, the current position of the axes in the activated coordinate system can be set to the values programmed for the axes. This is mainly used to calibrate the coordinate zero point to the current position of the axes. During this process, the current position of the programmed axes becomes the coordinate zero point.

A zero shift can be effected at the same time by the specification of a number not equal to zero, i.e. the axis is then positioned on the programmed point. Any zero points which might be saved for the specified axes are overwritten by G193.

Example

G193 X0 Z0	The current position of X and Z is becomes the new point of origin.
G193 X10	The zero point is set in such a way that the X-axis is set to the value ”10” after the shift.

3.2.2.70 G194 Program an additional basic offset

G194 is used to program a basic offset, which has an additional effect to the basic offset in the machine constants. The basic offset is shifted without consideration of the current coordinate system.

Syntax

G194 AXES

Meaning of the addresses

AXES	Axes, whose basic offset should be set.

Explanation

The basic offset programmed for G194 is added to the value from MK_GRUNDOFFSET and transmitted to the axes. In contrast to a coordinate shift in a coordinate system, G194 can also be used in an interrupt program.

The offset shift occurs directly in the axis computer, by-passing the interpreter and handler. The basic offset programmed with G194 is not saved and is lost when the control is switched off. The programmed values are stored from P928 in the parameter field.

Example

G194 X150	The basic offset of the X-axis is set to MK_GRUNDOFFSET+150.
G194 X0	The basic offset of the X-axis is reset to the value of MK_GRUNDOFFSET.

3.2.2.71 G195 Absolute coordinate shift of all S coordinate systems

With G195, all the S coordinate systems can be shifted by the programmed amount in reference to the zero points defined by G193/G92 at the same time. The shift has an additive effect to the current zero points of all S coordinate systems except for S0.

Syntax

G195 AXES

Meaning of the addresses

AXES	Axes, whose zero point should be shifted.

Explanation

G195 is used to program the offset of S0. Since all the other coordinate systems are relative to S0, this has the effect that all the other coordinate systems are shifted.

The machine constant MK_S0T0_VERSATZ_ERLAUBT can be used to select whether this shift should also be effective in S0. When the default setting is 0, a shift of S0 is not effective in S0, but rather only in all other coordinate systems. In this case, the total offset (P192ff) of the axes is represented as follows:

In S0: offset = Tm

In Sn: offset = S0+Sn+Tm with n > 0

Ifs MK_S0T0_VERSATZ_ERLAUBT is set to 1, a distinction is made in the creation of the total offset:

In S0: offset = S0+Tm

The shift with G195 is an absolute shift, i.e. the value programmed for the axes replaces the old offset. The shift is stored remanently in CMOS-RAM if the machine constant MK_NULLPUNKTE_SPEICHERN is set to 1. It is then also still effective after the control has been switched off or on.

Example

G195 X10	All coordinate systems except for S0 are shifted in X by 10 mm (MK_S0T0_VERSATZ_ERLAUBT = 0).
G195 X0	Shift of the X axis is canceled again.

3.2.2.72 G200 Geometry filter ”on”/”off”

With G200, a filter can be activated with which the noise of the programmed profile can be suppressed. Noise-infested profiles are usually created when the point density corresponds in size to the defined profile grid points.

Syntax

G200 AXES

Meaning of the addresses

AXES	Axes, which should be considered for the filter function with the specification of the filter constants for each axis.

Explanation

The geometry filter can replace several G1 blocks by a single G1. During this process, up to 20 grid points are saved and a line between the first and the last point is calculated with each new grid point. Then, the deviations between the calculated line and all saved grid points are determined and compared with the maximum allowed tolerances programmed for G200.

If the deviation in all points is within the tolerance, the new point is saved and the corresponding block removed from the input stream. Otherwise, the current block is saved and the previous optimum of saved grid points is output as the new block.

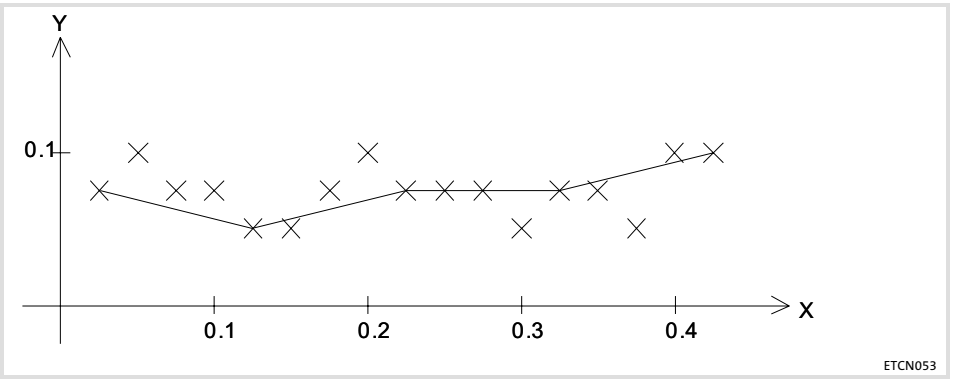
Only G1 blocks can be reduced, when nothing except for the axes specified for G200 is programmed. Generally, blocks with ”F”, ”M”, ”S”, ”T” or ”Q” cannot be reduced. The tolerances programmed for G200 must be specified in input units. The tolerances of the individual axes may be different in size and are considered based on each axis.

To be able to asses the efficiency of the geometry filter, the relation of reducible to reduced blocks is displayed in percent in P567. The maximum efficiency attainable is approximately 94 %. The time-based overhead of the geometry filter is fully compensated from an efficiency of approx. 15 % by the lower number of blocks to be executed.

The geometry filter is deactivated by programming G200 without axes.

Example

G200 X0.1 Y0.05	Suppress output of the profile until the total of profile pieces is exceeded in X by 0.1 mm or in Y by 0.05 mm.



3.2.2.73 G201 Change the acceleration and deceleration ramps

G201 can be used to program the acceleration and deceleration ramps in path operation and in route operation.

Syntax

G201 AXES J I

Meaning of the addresses

AXES	Axes, whose ramps should be changed to route operation.
J	For changing the ramps to path operation.
I	Which ramps should be changed 0: acceleration and deceleration ramp (default) 1: only acceleration ramp 2: only deceleration ramp

Explanation

To change the path ramps, the height of the acceleration and brake ramp is programmed in m/s2 for J. The entered value is limited to the value of MK_BAHNBEACHL or MK_BAHNBREMS in the machine constants.

The acceleration and deceleration ramps of the individual axes in route operation are programmed in m/s2 in the case of linear axes and in 1/s2 in the case of rotation axes, depending on the corresponding axis letters. The entered values are limited to the values of MK_BESCHL or MK_BREMS in the machine constants.

The letter I can be used to select whether only the acceleration ramps, only the deceleration ramps or both should be changed. If no I is programmed, both ramps are changed to the programmed value.

If G201 is programmed without parameters, the ramps on the path and of all configured axes are reset to the values set in the machine constants.

Example

G201 J1.5 I1	Change acceleration ramp on the path to 1.5 m/s^2.
G201 J2.5 I2	Change deceleration ramp on the path to 2.5 m/s^2.
G201	Reset all ramps to the values configured in the machine constants.

3.2.2.74 G209 Set the geometry counter

G209 is used to initialize the geometry counter to a user-specific value.

Syntax

G209 E

Meaning of the addresses

E	New value of the geometry counter (default = 0).

Explanation

The geometry counter is needed in the DIN display of the ETC-MMI for the synchronization of the progress display with the program processing in the NC control. It is set by the control to 0 at the start of the program and increased by 1 with every new geometry block.

In some cases, it might be useful to manipulate this geometry counter from the NC program. With G209, the geometry counter of the current NC channel can be set to a defined value.

The current value of the geometry counter is shown in the parameter field, namely in P565 for channel 0 and in P597 for channel 1.

Example

P2000=P565	flag current value of the geometry counter
...	Traverse blocks, which should not also be counted
G209 E=P2000	restore old value

3.2.2.75 G211 Transformation for two-axle articulated robot kinematics

G211 is used for switching on or off a transformation for a simple two-axle articulated robot kinematics (A,B) with support for an optional third axis (C) for the tool orientation.

Syntax

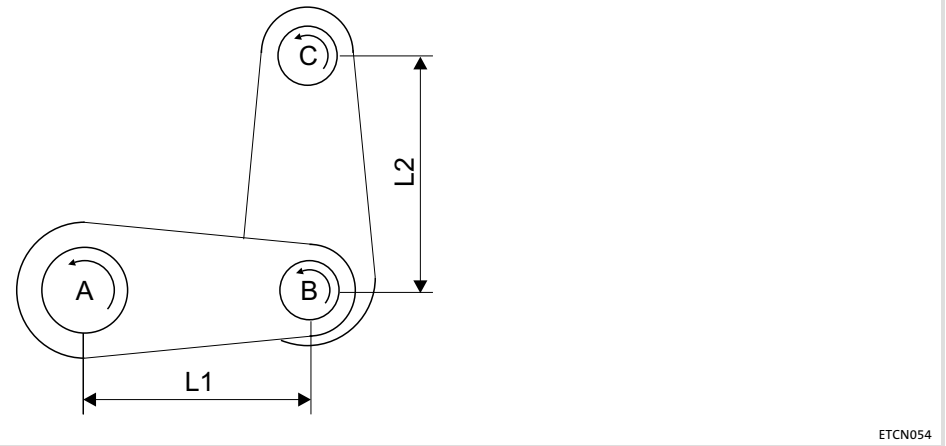
G211 E I J

Meaning of the addresses

E	For switching the optional third axis ”on” 0: without third axis 1: with third axis for tool orientation
I	Lever length L1, default value from P760 (MK_TECHNOLOGIEDATEN1[0])
J	Lever length L2, default value from P761 (MK_TECHNOLOGIEDATEN1[1])

Explanation

The function transforms from the X Y plane in the Cartesian coordinate system into a A B machine coordinate system. During this process, the tool orientation of the C axis, if desired (E1), is also transformed. The following figure shows the machine kinematics in the 0° setting of the A and B axis.



The lever lengths L1 and L2 may also be negative.

The X/Y coordinate of the pivotal point of the C axis is programmed when the transformation is activated. In the initial position A0, B0 this corresponds to the coordinate X0, Y0 in the Cartesian coordinate system.

During transformation, the two virtual axes X and Y are created, which are then used for programming in the Cartesian coordinate system. The maximum speed and the ramps of these axes are transferred from the MCs MK_BAHNVMAX, MK_BAHNBESCHL and MK_BAHNBREMS.

During transformation, the physical axes A and B can only still be moved indirectly via X,Y. During this process, the speed and acceleration of the physical axes which are involved in the transformation are not limited. This is especially important to note when the critical travel region of the machine is passed. Traveling with X/Y coordinates close to the pivotal point of the A axis will result in a very high traversing speed of the physical axes!

Two free application axis numbers are needed for the virtual axes. It is recommended that you keep free the first two axis numbers in the MCs for the virtual axes. In this case, X is assigned the axis number 0 and Y the axis number 1.

The transformation is deactivated by programming G211 without E or automatically at the end of the program or in the event of a program termination.

Example

G211 E1	Transformation with consideration of the C axis ”on”.
...	Profile with X,Y and C coordinates.
G211	Switch transformation off.
...	A,B coordinates are allowed from here again.

3.2.2.76 G226 Reconfigure hardware limit switch

G226 can be used to change the response of the control when traveling on the direction-dependent hardware limit switches of individual axes.

Syntax

G226 AXES

Meaning of the addresses

AXES	Identification of the axes, whose limit switch function should be changed. 0: deactivate limit switch 1: activate normal limit switch function 2: the limit switches effect a limit on the traversing range for the corresponding axis, but do not cause an error message or a program termination.

Explanation

The identification, which is programmed for the axes, defines how the control should respond to the limit switches of the corresponding axis. If the limit switches are deactivated, the control does not respond at all to the limit switches of the axis. In the case of the normal limit switch function, the control responds by stopping the traversing movement without delay and terminating a possibly running NC program.

If the identification 2 is programmed, the control responds to the limit switches, by stopping the traversing movement of the affected axis in this direction without delay and freezing the position of the axis. The position remains frozen until the axis should leave the frozen position again in the opposite direction. During the process, neither an error message is created nor the program is terminated. This function effects a cutting of profile areas when traveling on the limit switches.

To reactivate the default setting from the machine constants, G226 can be programmed without axes.

Example

N10 G226 X0	Limit switch of the X-axis ”off”.
N90 G226	Activate preset limit switch function for all axes.

3.2.2.77 G233 2D/3D axis correction

With 2D/3D axis correction, an axis is corrected in dependency on one or two axes via one of a maximum of 3 grid point tables.

Syntax

G233 AXE

Meaning of the addresses

AXE	Identification of the axis, which should be corrected. 0: correction tables for the specified axis ”off” 1: correction table 1 ”on” 2: correction table 2 ”on” 3: correction table 3 ”on”

Explanation

With this function, the position of an axis is corrected in dependency on one to two basis axes via a previously loaded table with equidistant grid points. In this case, the axis to be corrected may be one of the two basis axes. The correction takes place with the offset specified in the table. The interpolation between the grid points is linear.

The tool offset of the current tool Tn is calculated with G233 when the function is switched on and therefore must not already be considered in the loaded grid point table. The coordinates of the grid point table must be related to the reference coordinate system including the basic offset, i.e. to S0 without shift.

Up to 3 such correction tables can be loaded and be superimposed at the same time. The same target axis can be involved several times. If several correction tables are activated, the execution sequence of the corrections is important:

Table 1: correction of Y depending on X

Table 2: correction of Z depending on X, Y

First 1 then 2: The correction of Y is included in the correction of Z and the corrections are superimposed.

First 2 then 1: The corrections are independent and are not superimposed.

The sequence of the corrections to be executed corresponds to the programming sequence of the G233 functions.

The correction tables are deactivated for all axes by programming G233 without axis letters and at the end of the program or in the event of a program termination.

Example

G233 Z2	Correction table 2 ”on”, target axis in the table must be Z
G233 Y1	Correction table 1 ”on”, target axis must be Y
G233 Y3	Correction table 3 ”on”, target axis must be Y
	This defines the sequence: 2,1,3
G233 Z0	Correction of Z ”off”
	The remaining sequence is 1,3
G233 Y0	All correction tables of Y ”off”

Structure of correction file

The correction tables are created as binary files and must be transferred to the control either via a connected ETC-MMI or with the hyper terminal under the name ”ACHS3D??.KOR”. The ”?” must be replaced by any numbers or letters. The file consists of a 128 byte long header, a definition block and a block with the grid point values for the axis to be corrected.

Header:

The header is 128 bytes long and contains the null-terminated list of the axis letters of all involved axes, as well as the binary number of the correction table in the last byte. In this case, the first letter defines the axis, which should be corrected in dependency on the other axes.

0	1	2	3	4	127	128
	Z	X	Y	0	...	1-3

These are referred to as the correction axis and the other axes as basis axes in the following.

Definition block:

The definition block describes the range of validity and the size of the following grid point table. For each basis axis, it contains a FLOAT for the minimum and maximum value in input units and a ULONG for the number of grid points between the minimum and the maximum value. The total number of the grid points corresponds to the product of the number of the grid points of all basis axes.

0	4	8	12	16	20	24
	X_{min}	X_{max}	X_{anz}	Y_{min}	Y_{max}	Y_{anz}

Length = number of basis axes x 12

Grid points:

The correction values are defined in input units in table form each with a FLOAT for each grid point. In this case, the first grid point corresponds to the coordinate which the belongs to the minimum values of the basis axes and the last grid point of the coordinate which belongs to the maximum values of the basis axes.

0	4	8	12		4k
	Z_{11}	Z_{12}	Z_{13}	...	Z_{1k}

where k = Y_{anz}

0	4	8	12		4k
	Z_{21}	Z_{22}	Z_{23}	...	Z_{2k}

0	4	8	12		4k
	Z_{31}	Z_{32}	Z_{33}	...	Z_{3k}

0	4	8	12		4k
	Z_{i1}	Z_{i2}	Z_{i3}	...	Z_{ik}

where i = X_{anz}

Definition of the used data types

BYTE	unsigned 8-bit integer
ULONG	unsigned 32-bit integer, 4 bytes long, with the byte with the lowest value first
FLOAT	IEEE single precision floating point, 4 byte long, with the byte with the lowest value of the mantissa first.

3.2.2.78 G234 Direction-dependent height correction

In the case of direction-dependent height correction, the position of the specified axis is corrected dependent on the driving direction in the current plane via a grid point table.

Syntax

G234 AXE K

Meaning of the addresses

AXE	Identification of the axis, which should be corrected. 0: correction ”off” 1: correction ”on”
K	Parameter field index, from which the correction table is stored in the P-field.

Explanation

This function can be used, for example, to correct a Z axis by an offset dependent on the current driving direction in the XY plane. The correction is limited to a maximum value of +/-1 input units (e.g. mm or degrees).

With K, the parameter field index is programmed, from which the correction table is stored in the P field. The first value of the table must contain the number of the following table values. Every table entry contains a correction offset for a defined profile angle in the current (time of G234) selected plane. The table values must cover an angle range of 360°, the first table value applies to a profile angle of 0°. The table can have a maximum of 36 entries(10° per correction value). The interpolation between the entries is linear.

The correction is switched off at the end of the program or in the event of a program termination and by programming G234 without axis letters.

Example

P1200=4	Number of grid points
P1201=0.0	Offset for 0°
P1202=0.1	Offset for 90°
P1203=0.2	Offset for 180°
P1204=0.3	Offset for 270°
G234 Z1 K1200	Correction of the Z-axis ”on”
G234	Correction ”off”

3.2.2.79 G251 Accept step response of an axis

G251 is used to set the drive and attitude control parameters of an axis through the acceptance of a step response and can only be used by a special axis setting tool like the AXSCOPE. Generally, this function is not important for DIN programmers.

Syntax

G251 AXES I J K L

Meaning of the addresses

AXES	Speed stroke of the axis to be set in [mm/min] or [1/min]
I	Jump duration in seconds
J	Acceleration duration in seconds
K	Number of jumps
L	Identification 0: recording of the master axis (default) 1: recording of the slave axis 2: recording of master and slave axis

Explanation

The function oscillates the specified axis K-times with the specified speed, the accelerating duration J and the jump duration I to and fro. At the end of the movement the axis is positioned at its starting point again. The traverse path corresponds to I * speed. The pause between the jumps is I - J. If K is not specified, the axis oscillates to and fro 1-time.

During the output of the jump function on the axis, theoretical speed, actual speed, following error and correcting variable are recorded as a 16-bit integer data type and transmitted asynchronously to the PC. This can display the recorded information as a graph or evaluate it elsewhere.

If the identification L2 is specified for a synchronous axis, the control records the actual speed of the master axis, the slave axis, the synchronous distance between the two axes and the theoretical speed.

Example

G251 X1000 I0.2 J0.1	Output of a setpoint step-change with a speed stroke of 1 m/min, a jump duration of 200 ms and an accelerating duration of 100 ms.

3.2.2.80 G252 Value input via display device (see MMI)

The P field parameter specified in the G function is redetermined by means of an interactive user query.

Syntax

G252 A B C E X Y Z F

Meaning of the addresses

a	Allowed minimum value of the input parameter
b	Allowed maximum value of the input parameter
C	Index of the parameter to be entered in the parameter field of the control
E	Block number, which should be activated during termination of entry
F	Text number or a format statement in connection with ”..”
X	Column of the display
Y	Line of the display
Z	Character size for the display

Explanation

G252 is a function for interactively inputting a value via the control computer. The current value of the activated P field parameter is displayed as the default proposal for the input.

A lower limit can be preset under A and an upper limit under B for the value of the input. If one or both input limits are missing, the corresponding minimum or maximum values are used.

The address C holds the index of the parameter to be entered, in which the input value should be stored.

Under the address E, a block number can be programmed at which the program should continue if the input was canceled with the ESC button. When the input is completed with ENTER, the program is continued with the block following G252.

X can be used to select another column position and Y can be used to select another line position for the text output. In addition, an alternative character size can be defined under Z. If no X or Y is specified, the text appears at the position defined in the control computer in the defined size (application-dependent; standard: status line above softkeys in normal size).

During the input, the text specified under F="..." is output and the desired parameter accepted in the defined format. The input format and the display position of the parameter is to be indicated at the desired position in the text string with a format specification in accordance with the following rules:

%[flags][width][accuracy]type

Flags	
<nothing>	right-justified, leading blanks/zeros
-	left-justified, subsequent zeros
+	always output sign (including +)
' '	only issue neg. sign
#	for le, lE, lf : always output decimal point
for lg, lG:	Always output decimal point but without following zeros

Width	
0n	min n-positions, preceding with zeros
n	min n-positions, preceding with " "

Accuracy	
<nothing>	6 positions at le,lE,lf
.0	le,lE,lf do not output a decimal point
.n	max. n-decimal positions

Type	
lf	Floating point [-]dddd.ddd
le	Floating point [-]d.ddd e[±]ddd
lg	like le or lf; depending on accuracy automatic change-over of the display
lE	like le, with "E" before the exponent
lG	like lg, with "E" before the exponent
%	the % character should be displayed

Example

N10 G252 F="please enter value: %10.3lf [mm]" A0 C1024 P1024=500

The block N10 gets a new value for the parameter field 1024 (C) via the display. The value must not be entered smaller than 0 (A). In the display, the text appears at an internally defined position (not X and not Y).

Please enter value: 500 [mm]

The current value of P1204 is proposed as the input. The cursor (_) is positioned on the position furthest right to the input field, which has a width of 10 characters.

The input is completed when the ESC button or the ENTER button was pressed and the current input value is within the defined area boundaries. An input ended with ESC does not change the original value of the parameter field. The program continues at any rate (no E programmed) with the block after block N10.

3.2.2.81 G253 Output of a comment, optional with program termination

The text specified in the G function is displayed on the connected control computer. If an error number is specified, the current program can also be canceled.

Syntax

G253 A B C E X Y Z F ( )

Meaning of the addresses

a	First display parameter
b	Second display parameter
C	Third display parameter
E	error number to be displayed on the control computer
F	Text number or a format statement in connection with ”..”
X	Column of the display
Y	Line of the display
Z	Character size for the display

Explanation

G253 is a function for displaying a text or an error message at a predefined position and in a predefined size (application-dependent; standard: status line via softkeys in normal size) on the control computer. X can be used to select another column position and Y can be used to select another line position for the text output. In addition, an alternative character size can be defined under Z.

The first and most simple use of the function is to enter the text to be output in comment characters ”(”, ”)” in the same line as G253. It must be noted that no parameters are evaluated anymore until the end of the line after the comment.

The second possibility of displaying the text is to specify a text number under F. In this case, the control computer displays the text stored under the specified number.

The third variant also enables the inclusion of up to three numerical values into the text, which are specified under A, B and C. For this, the specification of the output text must be programmed under F= in quotation marks (”...”). In contrast to the comment, the format statement can appear at any position in the block. An additional comment is only interpreted as a comment if there is a format statement.

Note!

G253 is only allowed in the first NC channel. G253 is ignored in the block search.

The position and the format of the parameters to be inserted must be indicated at the desired position in the text string in accordance with the following rules.

%[flags][width][accuracy]type

Flags	
<nothing>	right-justified, leading blanks/zeros
-	left-justified, subsequent zeros
+	always issue sign (including +)
” ”	only issue neg. sign
#	for le, lE, lf : always output decimal point
for lg, lG:	Always output decimal point but without following zeros

Width	
0n	min n-positions, preceding with zeros
n	min n-positions, preceding with ” ”

Accuracy	
<nothing>	6 positions at le,lE,lf
.0	e,E,f do not output any decimal point
.n	max. n-decimal positions

Type	
lf	Floating point [-]dddd.ddd
le	Floating point [-]d.ddd e[±]ddd
lg	like le or lf; depending on accuracy automatic change-over of the display
lE	like le, with ”E” before the exponent
lG	like lg, with ”E” before the exponent
%	the % character should be displayed

As parameters, parameter field indexes can be assigned under A, B and C with indirect programming (A=P...), where the alphabetic sequence is also the sequence during conversion.

If the address E is programmed, an error handling graded according to the value programmed under E is started. Here, the text in the comment or anchored under F= text is output as the error text. The allowed values for E are 101 ... 499. The hundreds digit of the value programmed under E is interpreted as an ”error loading”. 2000 is added to the value specified under E, and displayed as an error message with the module identification 3 (interpreter) on the operator device.

If E is programmed, the specifications under X, Y and Z are ignored. If E is not programmed, no error handling takes place.

3.3 Formula processor

In the NC program, in addition to programming via G functions, it is also possible to input mathematical formula directly. A mathematical expression is indicated by a ":" at the beginning of the line. This can be preceded by a block number. Comments are allowed in a block with a mathematical expression when included in curly brackets "{}".

The formula processor can also be used to calculate complex expressions for the control at runtime. Both constants and elements from the parameter field can be used as operands and combined in any way. The arithmetic operations are indicated by agreed operations. The use of brackets is allowed.

3.3.1 Arithmetic operations

Syntax	Operation	Description
x + y	x + y	Addition
x - y	x – y	Subtraction
x * y	xy	Multiplication
x / y	x / y	Division
x ^ y	x^y	Power
SQR(x)	x^2	Square of a number
SQRT(x)	$\sqrt{x}$	Square root of a number
HYPOT(x,y)	$\sqrt{x^2 + Y^2}$	Amount of a vector in the plane
HYPOT3(x,y,z)	$\sqrt{X^2 + Y^2 + Z^2}$	Amount of a vector in the space
FMOD(x,y)	rest(*x/y*)	Remaining amount from division x / y
SIN(x)	sin*x*	Sinus of an angle in degrees
COS(x)	cos*x*	Cosinus of an angle in degrees
TAN(x)	tan*x*	Tangent of an angle in degrees
ASIN(x)	arcsin*x*	Arc sine, delivers an angle between -90° and +90°
ACOS(x)	arccos*x*	Arc cosine, delivers an angle between 0° and 180°
ATAN(x)	arctan*x*	Arc tangent, delivers an angle between -90° and +90°
ATAN2(y,x)	arctan(*y/x*)	Arc tangent with two arguments, delivers an angle between 0° and 360°
MOD360(x)	rest (*x*/360)	Angle x modulo 360°
ABS(x)	|*x*|	Absolute value of a number (without sign)
SIGN(x)	x/.x.	Sign of a number; +1 for x >= 0 and -1 for x < 0
ROUND(x)	round*x*	Round up or down a number to integer values
TRUNC(x)	trunc*x*	Delivers the positions before the decimal point of a number
FRAC(x)	x-truncx	Delivers the positions after the decimal point of a number
MIN(x,y)	min{*x,y*}	Delivers the smaller value
MAX(x,y)	max{*x,y*}	Delivers the larger value
RANDOM(x,y)	rand	Delivers random numbers between x and y

Syntax	Operation	Description
AND(x,y)	$(x \neq 0) \wedge (y \neq 0)$	Logical AND operation
OR(x,y)	$(x \neq 0) \vee (y \neq 0)$	Logical OR operation
XOR(x,y)	$(x \neq 0) \veebar (y \neq 0)$	Logical exclusive OR operation
NOT(x)	x = 0	Logical NOT
BITAND(x,y)	$x \wedge y$	Bit-wise AND operation
BITOR(x,y)	$x \vee y$	Bit-wise OR operation
BITXOR(x,y)	$x \veebar y$	Bit-wise exclusive OR operation
BITNOT(x)	$\overline{x}$	Bit-wise NOT
PI	π	Constant 3.141592654
PI180	$\pi/180$	Constant 0.01745329252

The arithmetic operations are processed in the sequence corresponding to the mathematical rules. To label leading positions, brackets up to a nesting depth of 4 levels are allowed.

Example

N10 : P1050 = sin(89.9 * P1051) / (34 + 91 * cos(P1052))
N20 : P1060 = atan2(P1,P0)
N30 : P1061 = hypot(P0,P1)

3.4 Block extensions

The blocks may be enhanced by one or several additional functions, which are indicated by the address letters **H, M**, **P**, **Q**, **S** and **T**. The functions may be programmed for individual blocks or for several blocks. They may also occur alone in a block.

If several functions are specified at the same time, they are processed in the fixed sequence **H, P**, **T**, **S**, **M**, **Q**.

3.4.1 Parameter assignment P

In every block, simple parameter assignments can be programmed directly, without G function or formula processor. However, for complex assignments containing arithmetic operations, the formula processor must be used.

The parameter field assignment can also be programmed several times within a block. This can be used, among other things, during subprogram calls in order to transfer several parameters. The parameter field assignment is always executed as the first operation in a block, independent of the position in the block at which it is programmed.

Possible assignment variants are:

P2000=17: The number 17 is stored in parameter 2000.

P2000=P0: The contents of P0 are assigned to parameter 2000.

PP1500=3.5: The value 3.5 is assigned to parameter, whose index is specified in P1500.

P2000=PP10: The contents of the parameter are assigned to P2000, whose index is entered in P10.

PP10=PPP11: The value of the parameter field, whose index is contained in the parameter field, whose index is stored in P11, is assigned to the parameter, whose index is displayed in P10.

Example

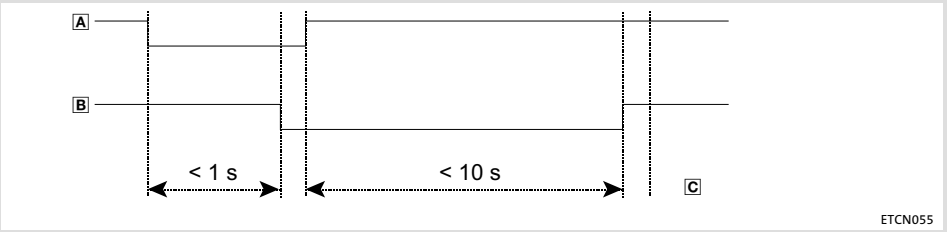
```
N10 G0 X3.14 P4000=0 P2000=P0
P3000=PP200 P3100=123.2 P3333=P100
G22 L7999 P2000=1 P2001=P1
```

3.4.2 H functions

H functions are provided for changing technology parameters. The number programmed for H is transferred to the PLC and must be evaluated there.

The H function is always evaluated as the first function in the block. Generally, H functions are generally time synchronized, i.e. the following function is only interpreted if the transfer by the PLC was acknowledged.

The H functions are transmitted to the PLC by means of handshake with strobe and acknowledgement. The NC signalizes a new H function to the PLC with the negative edge of the strobe.



A Strobe
B Acknowledgment
C Next block

If the transmission of the technology parameters lasts longer than 10 seconds, the PLC must also remove the read enable, otherwise the control will report the error message ”Time Out bei der H-Funktionsübergabe an die SPS” (Time Out during the H function transfer to the PLC).

3.4.3 M functions

M functions are transmitted to the PLC. These must be evaluated there and the desired function triggered. A distinction is made between ”synchronized” and ”non-synchronized” M functions.

The M functions with the numbers 0 ... 499 are synchronized. If these are transmitted to the PLC, the processing of the NC program is interrupted until the PLC acknowledgement has arrived and the read enable has been set.

In the case of non-synchronized M functions (>= 500), the next block of the NC program is executed immediately after the output without waiting for an acknowledgement from the PLC. It must be noted that the speed is also reduced between two successive traverse blocks when exact positioning is deactivated if several M functions are programmed in succession without a traverse movement.

The following M functions are also evaluated in the NC computer:

M30: Program end

M14: Switches from the summation of the path length to P551 (profile)

M15, M16, M21: Switches from the summation of the path length to P551 (empty run)

M48: Speed overlap enabled (override on)

M49: Speed overlap disabled (override off)

M1014: Like M14, however as an asynchronous M function

M1015: Like M15, however as an asynchronous M function

M1048: Like M48, however as an asynchronous M function

M1049: Like M49, however as an asynchronous M function

Using a M function, it is also possible to initiate a subprogram call. For this, all the M functions to be expanded must be entered in the machine constants, under the identifier **MK_MFKT_UPR_TABELLE**,.

If the control encounters a corresponding M function in the course of program processing, it will create a subprogram call with the program number ”9000 + M function number” after(!) all the other information from this block has been processed.

M function numbers from 1 to 255 can be used for this type of subprogram call.

The M function itself is transmitted normally to the PLC. If the same M function is programmed in the subprogram that was called, it does not result in the subprogram call being called again.

Stop!

If another subprogram is called in the subprogram 9000(+M function number), which contains the triggering M function, a call recursion occurs and then the error message ”Program stack overflow”

This subprogram mechanism is not only active in the automatic program processing. The same M function is also expanded as an individual block in a subprogram call.

Example

N10 G0 X300 M14
N20 G2....

After the control has traversed the X-axis to the value 300, and the information has been transmitted via the M14 to the PLC, the subprogram %9014 is executed. Processing with block N20 is only continued after subprogram %9014 has been completed.

The program sequence of the example is functionally identical with

N10 G0 X300 M14
G22 L9014
N20 G2....

3.4.4 Q functions

States of CAN I/O modules can be programmed under the address ”Q”. Q functions are also referred to as ”fast inputs”, because they are entered directly in the course interpolator, without a detour via the PLC, into the NC program.

Since only CAN outputs are available for the digital outputs, these are always connected to the PLC cycle.

The PLC cannot change the initial state of the outputs concerned anymore, until the enable has been reset by the PLC. However, it can read the current initial state at any time independent of this.

”Fast outputs” are not automatically reset by the control after the end of the program or in the event of a termination. If necessary, this is performed by the PLC by withdrawing the enable and resetting the outputs.

For more information on enabling the “fast inputs/outputs”, refer to the documentation MC and NC software manual.

Q-statements may also be programmed more than once within a block.

Example

Q0=0 Q14=0 Q15=1	Initialize states of the outputs
G60 X0	Exact positioning ”off”
G0 X159 Y31 Q0=1	Approach position 159 in X and 31 in Y, set output 0 to 1 and
G1 X163 Y77	execute the next block without stopping
G1 Y121 Q14=1 Q15=0	Continue traveling and at the end of the block, set output 14 to 1 and
G1 Y154	set output 15 to 0

3.4.5 S functions

The letter S in the DIN block always stands for the specification of the current workpiece coordinate system, to which the following position specifications refer. Under a coordinate system, the control understands the definition of a zero point offset for each axis (related to the zero point in S0), where S0 is related to the machine zero point (home position + basic offset).

In ETCXC, 100 workpiece coordinate systems of this kind are available. These are selected with S0 ... S99.

If a 1 is entered in the machine constant MK_NULLPUNKTE_SPEICHERN, the currently switched on S coordinate system and the zero point offsets of all coordinate systems are saved and restored after the control has been switched on. Otherwise, the control is in S0 after it has been switched on and the offsets of all coordinate systems are zero.

For all workpiece coordinate systems, the zero point can be shifted or set using the functions G92, G93 and G193. S0 can only be shifted with these functions if machine constant MK_S0T0_VERSATZ_ERLAUBT is set to 1. If S0 is shifted, it must be noted in this case that all the other coordinate systems are also shifted!

Function G195 is used especially to shift all coordinate systems in which the offset of S0 is shifted. It is also allowed if the above-mentioned machine constant is set to 0. However, the offset of S0 is then not effective in S0 but in all other coordinate systems.

Function G54 is used to temporarily shift the zero point of the current S coordinate system. The shift remains active until it is deactivated with G53 or the program is ended. The shift by G54 is not saved remanently.

During the change over to another coordinate system, the current actual positions in S_{old} are cleared with the zero point offsets of the new system (S_{new}) and displayed.

The offsets of the workpiece coordinate systems can be loaded via the ETC-MMI into the control or transferred from the control into the PC.

Example

G0 X0 Y0 S0	In the coordinate system S0, travel to the position X0, Y0.
G92 X0.1 S1	Change over to the coordinate system S1 and shift the zero point of the X-axis about 0,1 mm.
G1 X100 F2500	Profile is S1 withdraw
M6 S=P711	Change over to the coordinate system, whose number is entered in parameter P711 and then output the M function M6.

3.4.6 T functions

The letter T in the DIN block always stands for the specification of the current tool coordinate system, to which following position specifications relate. Under a coordinate system, the control understands the definition of a zero point offset for each axis.

In ETCXC, 100 such tool coordinate systems are available, which are selected with T0 ... T99.

In addition to the offsets of the axes, a table containing 30 tool data entries, which are shown in parameter field from P700 for the current T, is assigned to every T-coordinate system. These tool data entries can be used within a tool management in order to save tool-specific parameters. These are used during the tool radius compensation.

If a 1 is entered in the machine constant MK_NULLPUNKTE_SPEICHERN, the currently activated T coordinate system, the zero point offsets and the tool data of all coordinate systems are saved and restored after the control has been switched on. Otherwise, the control is in T0 after the switch on and the offsets of all coordinate systems are zero.

In all tool coordinate systems, the zero point can be shifted or set using the functions G92, G93 and G193.

During the change over to another coordinate system, the current actual positions in T_{old} are cleared with the zero point offsets of the new system (T_{new}) and displayed.

The tool data and the offsets of the tool coordinate systems can be loaded via the ETC-MMI into the control or transferred from the control into the PC.

Determination of the total offset of the S and T coordinate systems (P192ff)

MK_S0T0_VERSATZ_ERLAUBT = 0	
In S0: offset = Tm	
In Sn: offset = S0+Sn+Tm	with n > 0
MK_S0T0_VERSATZ_ERLAUBT = 1	
in S0: offset = S0+Tm	
In Sn: offset = S0+Sn+Tm	with n > 0

Example

G193 X0 Y0 T1	Set current position of the X and Y axis in T1 to zero.
T5 S=P711	Change over to T5 and then activate the S coordinate system, which is entered in the tool data of T5 in P711.

3.5 Data fields

3.5.1 P field

For programming with variables, a data field (the parameter or P field) is available in the control. This data field contains different data:

- Internal data of the NC computer

  This data is created by the processing of a program and/or reflects certain states of the NC computer. Only read accesses can be made to this data. The access to most of this data is time synchronized.

- Data for cycles and macros

  This area is reserved for use in cycles and macros. In this area, data is transferred from the NC computer to the cycles. In addition, it is used as a variable area for the cycle programs. Cycles and macros have a write and write access to this data, user programs should only have read access to this area. The access is not time synchronized.

- Data for user programs

  This area is provided for free use in user programs. These have write and read access. The access is not time synchronized. The meaning of the parameters is program-dependent.

The parameter field has a default size of 2048 entries; these can be enlarged with the machine constant MK_PFELD_GROESSE user-specifically.

Distribution

0 1023	Internal data of the NC computer
1024 1499	Data for cycles
1500 2047	User data
2048 MK_PFELD_GROESSE	Enhanced P field

The parameter field can be accessed by

- G functions
- Arithmetic functions (formula processor)
- Messages of the MMI
- Messages of the PLC

Assignment of system parameters

The assignment of the area of the parameter field, in which the internal data of the NC computer is stored, is specified in the following. The specified digits correspond to the parameter numbers under which the variables are addressed. Axis-related values are always entered in the sequence in which they are defined in the machine constant MK_APPLACHSIDX.

Meaning of the units in the following tables:

Unit	Meaning
Inc	Increments
°	Degree
Unit	Translative path input unit, machine constant-dependent (default = mm)
mm	Millimeter
m	Meter
h	Hour
min	Minute
s	Second
ms	Millisecond
GIT	Coarse interpolation cycle
MC	Unit is dependent on the machine constants
%	The parameter is specified in percent
---	The parameter has no unit

With some parameters, the unit is dependent on whether the parameter belongs to a linear axis or a rotation axis. In the case of linear axes, the input unit is also dependent on the value in the machine constants MK_METRISCH, MK_CONST_REL_MM and MK_CONST_REL_INCH. The following enhanced table applies to these parameters.

Unit	Meaning with linear axes	Meaning with rotation axes
mm *	Input unit	Degree
mm/min *	Input units/minute	Revolutions/minute
m/s² *	Meter/second²	Revolutions/second²
Inc/mm *	Increments/input unit	Increments/degree

A cross in field ”Sync” indicates that the access to the corresponding parameter is time synchronized, i.e. that the contents of the parameter in a NC program is only read when the block was executed before. The further interpretation of the NC program is also stopped until this has happened.

Axis-specific parameters

Index	Meaning	Sync	Unit
0 ... 15	NC actual position of the axes in reference to the current coordinate system	x	mm *
16 ... 31	Total of modal actual position and NC actual position of the axes (P160ff+P0ff)	x	mm *
32 ... 47	NC target position of the axes in reference to the current coordinate system	x	mm *
64 ... 79	Zero point offset of coordinate system S0	x	mm *
80 ... 95	Zero point offset of coordinate system Sn (-> P586) from channel N	x	mm *
96 ... 111	Zero point offset of coordinate system Sn (-> P554) from channel 0	x	mm *

Index	Meaning	Sync	Unit
112 ... 127	Zero point offset of coordinate system Tn (-> P585) from channel N	x	mm *
128 ... 143	Zero point offset of coordinate system Tn (-> P553) from channel 0	x	mm *
144 ... 159	Modal target position of the axes in reference to the current coordinate system	x	mm *
160 ... 175	Modal actual position of the axes in reference to the current coordinate system	x	mm *
176 ... 191	Modal target position specification in reference to the current coordinate system	x	mm *
192 ... 207	NC total offset of the axes of S0+Sn+Tn (channel-independent)	x	mm *
208 ... 223	Modal speed specification of the axes	x	mm/min *
224 ... 239	Real actual position of the axes related to the basic offset	x	Inc
240 ... 255	Modal acceleration specification of the axes	x	m/s^2 *
256 ... 271	Tolerance margin of the axes	x	Inc
272 ... 287	Modal actual speed of the axes	x	mm/min *
304 ... 319	Negative software traverse limits of the axes related to the basic offset	x	mm *
320 ... 335	Real actual position of the axes related to the basic offset	x	mm *
336 ... 351	Positive software traverse limits of the axes related to the basic offset	x	mm *
352 ... 367	Saved actual positions of the axes related to the basic offset, these are saved by an external trigger signal at the same time	x	mm *
394 ... 409	States of the input signals of the axes (limit switch, reference cam,...) Bit0: positive limit switch Bit1: negative limit switch Bit2: reference cam Bit3: reserve input	x	---
410 ... 425	States of the input signals of the slave axes for synchronous axes. Bit0: positive limit switch Bit1: negative limit switch Bit2: reference cam Bit3: reserve input	x	---
426 ... 441	Synchronous distance between master and slave axes for synchronous axes	x	mm *
442 ... 457	Min. synchronous distance, determined via 200 fine interpolation cycles	x	mm *
458 ... 473	Max. synchronous distance, determined via 200 fine interpolation cycles	x	mm *
480 ... 495	Conversion factor of input units in increments	x	Inc/mm *
496 ... 511	Basic offset from the MCs (MK_GRUNDOFFSET)	x	mm *
640 ... 655	Default speed of the axes for use in cycles, initialized to the value of the maximum speed in MK_VMAX	x	mm/min *
928 ... 943	Additional basic offset of the axes (G194)	x	mm *
944 ... 959	Desired positions of the axes including all the transformations and coordinate system offsets	x	mm *
960 ... 975	Extension factors of the axes (G75)		---
992 ... 1007	Programmed target position from the last interpreted DIN block		mm *

Parameters for program management

Index	Meaning	Sync	Unit
512,513	Program number and block number of the starting NC program in start mode 0 (MMI)	x	---
514,515	Program number and block number of the starting NC program in start mode 1 (PLC)	x	---
516,517	Program number and block number of the starting NC program in the start mode 2 (ZYK)	x	---
518,519	Program number and block number of the starting NC program in start mode 3 (APP1)	x	---
520,521	Program number and block number of the starting NC program in the start mode 4 (APP2)	x	---
522,523	Program number and block number of the starting NC program in start mode 5 (APP3)	x	---
524,525	Program number and block number of the starting NC program in start mode 6 (APP4)	x	---
526,527	Program number and block number of the starting NC program in start mode 7 (APP5)	x	---
528	Program number as a trigger condition for the block search function	x	---
529	Block number as a trigger condition for the block search function	x	---
530	Logical block number as a trigger condition for the block search function	x	---
531	M function as a trigger condition for the block search function	x	---
532	Number of loops passed as a trigger condition for the block search function	x	---
533	Path length, about which it should be positioned into the target block within the block search function	x	---
534	Line number as a trigger condition for the block search function	x	
535	Tool number as a trigger condition for the block search function	x	
536	Geometry number as a trigger condition for the block search function	x	
537	Position in the target block as a measurement between 0 and 1 for the block search function. 0: block start 1: block end	x	
540	Bit coded display, which signalizes whether the oscillation function (G36) is switched on for an axis	x	---
541	Shows whether the buffered positions in P352 ... 367 are valid. They are valid if the trigger signal was recognized. This parameter must first be reset to 0 to enable the positions to be latched again.	X	---
544	Number of the NC channel for which the subsequent data is valid.	x	---
545	Current program number	x	---
546	Current block number	x	---
547	Current logical block number	x	---
551	Total of all executed profile pieces without empty runs since the start of the program	x	---
552	Covered path length in the current block	x	---
553	Current tool coordinate system Tn	x	---
554	Current tool coordinate system Sn	x	---
555	Current path speed	x	mm/GIT *

Index	Meaning	Sync	Unit
556	Current processing state of channel 0 for diagnosis purposes 0: Idle 1: Run 2: Brake termination 3: Termination wait quit 4: Termination 5: Brake interrupt 6: Interrupt 7: Brake block jump 8: Block jump 9: Brake interrupt 10: Interrupt 11: Brake error 12: Error 13: Balancing run	x	
557	Current increment, e.g. stitch length during sewing	x	mm *
558	Current interpreter state 0: No program active 1: Single function active 2: Program active 3: Interrupt program active 4: Block search active	x	---
559	Bit coded state of modal functions Bit0: 0=route operation (G30), 1=path operation (G31) Bit1: 0=empty run (M15,M16,M21), 1=profile (M14) Bit2: 0=event handler disabled (G132), 1=event handler enabled (G133) Bit3: 0=override enabled (M48), 1=override disabled (M49) Bit4: 1=G114 active Bit5: 1=G231 active	x	---
560	Current line number (not block number) in the NC program	x	---
561	Current setpoint speed, e.g. for a spindle. Corresponds to the value programmed for address E in the DIN block (G0-G3).	x	---
562	Number of the M function, by which the last subprogram call was triggered. For use in cycles, which are called by M functions (MK_MFKT_UPR_TABELLE).		---
563	Last output H function number	x	---
564	Current number of preprocessed blocks in the block search buffer		---
565	Geometry counter for block progress display	x	---
566	Progress display for current traversing block: 0=block start, 1=block end	x	---
567	Efficiency of geometry filter G200 in %	x	%
576	Number of the NC channel, for which the subsequent data is valid.	x	---
577	Current program number	x	---
578	Current block number	x	---
579	Current logical block number	x	---
583	Total of all executed profile pieces without empty runs since the start of the program	x	---
584	Covered path length in the current block	x	---
585	Current tool coordinate system Tn	x	---
586	Current tool coordinate system Sn	x	---
587	Current path speed	x	mm/GIT *

Index	Meaning	Sync	Unit
588	Current processing state of channel N for diagnosis purposes 0: Idle 1: Run 2: Brake termination 3: Termination wait quit 4: Termination 5: Brake interrupt 6: Interrupt 7: Brake block jump 8: Block jump 9: Brake interrupt 10: Interrupt 11: Brake error 12: Error 13: Balancing run	x	
589	Current increment, e.g. stitch length during sewing	x	mm *
590	Current interpreter state 0: No program active 1: Single function active 2: Program active 3: Interrupt program active	x	---
591	Bit coded state of modal functions Bit0: 0=route operation (G30), 1=path operation (G31) Bit1: 0=empty run (M15,M16,M21), 1=profile (M14) Bit2: 0=event handler disabled (G132), 1=event handler enabled (G133) Bit3: 0=override enabled (M48), 1=override disabled (M49)	x	---
592	Current line number (not block number) in the NC program	x	---
593	Current setpoint speed, e.g. for a spindle. Corresponds to the value programmed for address E in the DIN block (G0-G3).	x	---
594	Number of the M function, by which the last subprogram call was triggered. For use in cycles, which are called by M functions (MK_MFKT_UPR_TABELLE).		---
595	Last output H function number	x	---
596	Current number of preprocessed blocks in the block search buffer		---
597	Geometry counter for block progress display	x	---
598	Progress display for current traversing block: 0: Block start 1: Block end	x	---
599	Efficiency of geometry filter G200 in %	x	%
610	Size of the free memory for managing NC programs		Byte
611	Size of the free memory for managing symbolic program numbers		Byte
612	Number of pending and not yet confirmed error messages		
613	Value of MK_S_VERSATZSPERRE from the machine constants		
666 ... 669	Current value of override 0 ... 3		
670	Maximum value of the acceleration ramp on the path, see G201		m/s^2
671	Maximum value of the deceleration ramp on the path, see G201		m/s^2
672	Speed preselection for modal travel	x	Unit/min
673	Increment specification for step travel		Unit

Index	Meaning	Sync	Unit
674	Conversion factor of input units in mm		mm/Einh
675	Conversion factor of m/min by input units/GIT	x	Unit/GIT m/min
676	Conversion factor of input units/min in input units/GIT	x	min/GIT
677	Conversion factor of m/s^2 in input units/GIT^2	x	Unit/GIT^2 m/s^2
678	Conversion factor of input units/s^2 in input units/GIT^2	x	s^2/GIT^2

Technology-specific parameter

Index	Meaning	Sync	Unit
288 ... 303	Analogue measured value of channel 0 ... 15 (-> MK_MESS_AUFLOESUNG)	x	MC
368 ... 373	Axis positions, determined by measuring cycles, for a basic rotation	x	mm *
374 ... 376	Basic rotation angle A, B, C in the YZ, ZX and XY plane	x	°
685	Thread pitch (G33)	x	mm *
686	Thread pitch correction (G33)	x	mm *
687	Thread cylinder correction (G33)	x	mm *
688	Thread measurement distance (G33)	x	mm *
690	Radius in the polar machine coordinate system (MK_MASCH_POL...)	x	mm *
691	Angle in the polar machine coordinate system	x	°
692	Angle of the C-axis in the polar coordinate system	x	°
696	Angle of the oblique X-axis from MK_X_WINKEL	x	°
697	Sinus of the angle of the oblique X-axis	x	---
698	Cosinus of the angle of the oblique X-axis	x	---
699	Tangent of the angle of the oblique X-axis	x	---
700 ... 725	Tool data (Tn) of channel 0, P700 = tool radius [mm]	x	---
730 ... 755	Tool data of channel N (-> P576) P730 = tool radius [mm]	x	---
760 .. 769	Technology data from MK_TECHNOLOGIEDATEN1		---
770 ... 779	Technology data from MK_TECHNOLOGIEDATEN2		---
780 ... 789	Technology data from MK_TECHNOLOGIEDATEN3		---
790 ... 799	Technology data from MK_TECHNOLOGIEDATEN4		---

Parameter for time recording

Index	Meaning	Sync	Unit
377	Total runtime since the start of the NC program	x	GIT
378	Total of the travel times since the start of the NC program	x	GIT
379	Total of waiting times for M functions since start of the NC program	x	GIT
380	Time since the control was switched on	x	s
381	Time since the control was switched on	x	min
382	Time since the control was switched on	x	h
389	Total of the dwell times since the start of the NC program	x	ms
390	Total runtime since the start of the NC program	x	ms
391	Total of the travel times since the start of the NC program	x	ms
392	Total of waiting times for M functions since start of the NC program	x	ms
393	Total of the waiting times by Q-Tab and V-Tab comparisons since the start of the NC program	x	ms

Technology-specific user parameters

The parameters in the range between P1024 and P1099 are provided for technology-specific functions and only of importance for the respective technology. For all of the technologies not listed here, parameters P1024 ... P1099 are available to the user.

Material cutting

Index	Meaning	Sync	Unit
1024	Overcut length		mm *
1025	Cutter length		mm *
1026	Total cutter abrasion		mm *
1027	Minimum length of an empty run for lifting the holding-down device		mm *
1028	Copy from MK_KONTURWINKEL2		°
1030	Advance-before plunge		mm *
1032	Min. cutter oscillation speed (G97: spindle type 2)		1/min
1033	Path speed for min. cutter oscillation		mm/min *
1034	Max. cutter oscillation speed		1/min
1035	Path speed for max. cutter oscillation		mm/min *
1036	Path distance window as a cutter criterion for the cutter grinding for the customer-specific comparison operation type 0 (G130 V0)		mm *

Polystyrene cutter

Index	Meaning	Sync	Unit
1040	Dwell time for non-tangential (not equal to 180°) profile transitions		s
1041	Preliminary position of the resistance wire in X		mm *
1042	Max. reached position of the X axis before the standstill of X		mm *
1043	Default position of the X-axis, at which the resistance wire is switched on		mm *

Axis positioning handler for handling tasks (G97 X10)

Index	Meaning	Sync	Unit
1040	Intended speed for axis positioning		mm/min *
1041	Target speed on reaching the target point		mm/min *
1042	Intended traverse path after triggering		mm *
1043	Accumulated actual traverse path after triggering		mm *
1044	Number of the input, which should be used as a trigger signal (0 ... 23)		---
1045	Positioning state 0: Standstill, 1: Find trigger signal, 2: Positioning running, 3: Intended traverse path reached or exceeded (P1041 != 0), 4: Intended traverse path reached, standstill (P1041 == 0)		---
1046	Acceleration and brake ramp		m/s^2 *
1047	Current default speed on the positioning axis		mm/min *

Sewing (G97 X5|X6)

Index	Meaning	Sync	Unit
1050	Y-correction value which is added for each mm of feed of the X-axis to the position of the Y-axis. Only in the case of spindle type 6		---
1051	Number of stitches of sewing head 1 since the control was switched on		---
1052	Number of stitches of sewing head 2 since the control was switched on, only in the case of spindle type 5 and two sewing heads		---
1057	Speed specification from PLC to spindle handler for speed reduction during sewing.		1/min
1058	Path target specification for customer-specific comparative operation of type 2		mm *
1059	Corner speed of the sewing motor in percent of the setpoint speed, for reducing the speed for non-tangential block transitions		%

Rectraction handler for measurement machine

Index	Meaning	Sync	Unit
1060	This is set when a probing has happened. It can be used in the probe cycle as a condition for a modal branch (G130)		---
1061	Number of error triggered for diagnosis purposes		---
1062	Probe path, reserved for probe cycle		mm *
1063	Probe path tolerance, reserved for probe cycle		mm *
1064	Probe speed, reserved for probe cycle		mm/min *
1065	Deceleration ramp to stop the traverse movement after a probe		m/s^2 *
1066	Rectraction speed after probing		mm/min *
1067	Rectraction acceleration after probing		m/s^2 *
1068	Rectraction path relative to the probing position		mm *
1069	Waiting time after probing triggering for filtering out triggered errors		ms
1070	Probe status 0: no probing taken place yet 1: valid probing taking place 2: probing could be valid, decision in the next GIT		
1071 ... 1086	Saved probing positions. For the pos. edge these are accepted at the trigger signal.		mm *

3.5.2 Q field

The Q field is a bit-oriented field with a size of 64 bits. The Q field can be used to control the progress of a NC program (e.g. G150) by means of external events. The individual bits can be controlled by the PLC in data block 1. The assignment can be freely defined by the user.

If fast inputs are configured (see MC and NC software manual), the corresponding Q bits are not read from data block 1 of the PLC, but rather directly from the corresponding inputs.

4 Machine constants

This chapter describes the machine constants (MCs) of the controls ETCPC and ETCHC. Some of the described MCs are not available in both control types due to the different hardware features. The respective identifiers are marked with a corresponding footnote.

4.1 Basics

MCs are used to adapt the control to the specific field of application. These include the actual machine with its axes as well as the fields of technology and operating philosophy. Therefore, the extent to which the end customers and users can affect the MCs should be limited.

At the end of this chapter, you can find a complete overview of the MCs including comments. This overview is included in the supply of each ETC-MMI under the name MUSTER.MK (SAMPLE.MC), but can also be supplied separately.

Note!

The file ”MUSTER.MK” (”SAMPLE.MC”) serves as a basis for your specific adaptations. Therefore, it is recommended to generate a copy of the file. The name of the file is irrelevant. It must be provided with the extension ”.MK”. However, it is recommended to use a name that is related to the machine, e.g. the machine number.

The file can be transferred to the control in two different ways. Either via an ETC-MMI whenever the control is restarted or by means of any terminal program via the monitor interface of the control. In the first case, the file is stored on the hard disk of the PC. The second case is only necessary if no PC-supported MMI is connected to the control.

The MCs are stored in the control in a non-volatile memory. This is where they are updated if the MCs are changed in any way. Controls delivered ex works do not contain an MC file. Therefore, the message ”keine Maschinenkonstantendatei geladen” (”no machine constants file loaded”) appears when the control is first put into operation.

When adapting the MC file, please make sure that MCs which are not included in the MC file are not reset to the default value, but keep the previously set value. However, to gain a better overview, all MCs should be described in the MC file. Some MCs have a limited value range. If the entered value lies outside this value range, it will be limited to this range.

The exact name of the MCs must be entered, otherwise the NC computer will not recognize it. During the transfer of the MCs, the ETC issues an error message for each MC it does not recognize. From the moment the transfer has been completed without errors, the MCs are valid in the machine. In general, it is always possible to transfer the MCs. However, MCs that affect the storage distribution within the control only become effective after the control has been restarted again. All of the other MCs become effective immediately after the transfer or at the latest when the control is ready to execute a program.

The MCs can be divided in specific interrelated fields. However, the order is optional.

4.2 Test settings

The test settings are used for operating or testing the ETC without the machine or to switch off specific parts of the functional range (PLC).

4.2.1 MK_TEST_OHNEMECHANIK

This machine constant is used for testing the control functions without having to connect the machine. In the process, the control switches the actual value encoder inputs to simulation operation. Correcting variables are still output. If there are drives that are already connected, switch off the servo amplifier. Otherwise, this may result in unwanted traversing movements!

Value	Meaning
0	The drives and position measurement systems must be connected properly (default).
1	To test MMI, PLC and DIN programs while the drives are disconnected.

4.2.2 MK_SPS_DUMMY

This machine constant switches the integrated PLC off or on.

Value	Meaning
0	The loaded PLC is executed automatically (default). If this MC is at 0 when the control is started up and no PLC is loaded in Flash-PROM, an error message is output.
1	All PLC enables required for the operation of the control are set. M functions do not take effect and all inputs and outputs remain unused. The loaded PLC is not executed. However, no error message is output if no PLC is loaded.
-1	The PLC is not executed and all enables are reset. Therefore, it is not possible to start a program or to traverse axes. No error message is output if no PLC is loaded.

Note!

When transferring the MCs from an ETC-MMI via the Dual-Port-RAM interface, the value -1 is always entered for this MC in Flash-PROM. This is to prevent the PLC from starting to soon when the control is restarted again. Nonetheless, the value in RAM, which has been transferred from MMI, is kept.

After a PLC has been transferred via the programming interface **CodeSys**, this MC is automatically set to 0 in RAM and the PLC is started. The value in Flash-PROM remains unchanged.

4.3 Software configuration

4.3.1 MK_KUNDE

This machine constant is a string constant by means of which the different customer-specific extensions are activated within the control. This includes technology-specific correction modules and M function handler.

Value	Meaning
""	Standard TRC, no special treatment of M functions (default).
"SCHNEIDEN" (CUT)	TRC with tangential correction of the C axis for cutting technology, special treatment for M14, M15, M16 und M1014, M1015, M1016.
"SCHNEIDEN1" (CUT1)	Like "SCHNEIDEN" (CUT), but with additional functions for the management of the knife length at the beginning and the end of the profile, additional special treatment for M46, M47, M50, M51 and M1046, M1047, M1050, M1051.
"SCHNEIDEN2" (CUT2)	TRC compatibility mode with tangential correction of the C axis for cutting technology without special treatment of M functions.
"SCHNEIDEN3" (CUT3)	Like "SCHNEIDEN2" (CUT2), but an additional rectangular approach or withdrawal block is inserted for G41/42 or G40.
...	Customer-specific modules that cannot be listed here.

4.3.2 MK_VOREINSTELLUNG

This machine constant is a string constant via which several default settings of G functions can be implemented. The following G functions may be specified within the string separated by a semicolon:

Value	Meaning
G0-G3 G100-G103	Set modal G function, optionally with speed specification F Example: AG1 F1000
G17 / G18 / G19	Selection of main plane X-Y / Z-X / Y-Z
G26	Activation of software limit switch of the specified axes, example: AG26 X0 Y0
G30 / G31	Activation of route operation / path operation
G60	Activation of Look Ahead or exact positioning, example: AG60 X0
G90 / G91	Activation of absolute dimension / incremental dimension for all axes
G112	Activation of tangential correction of the specified axis (A,B,C,a,b,c), example: AG112 C1
G122	Configuration of traverse button operation, example: AG122 X1

The specified string must be specified in inverted commas and must not exceed 80 characters. The configured default settings are automatically activated when the control is started up after the MCs have been loaded as well as at the program end and after a program termination.

Example

MK_VOREINSTELLUNG "G17; G1 F30000; G26 X0 Z0";

For further information on the G functions, refer to chapter "CNC programming".

4.3.3 MK_NCPROG_OHNE_KOMMENTARE

This machine constant suppresses the storage of comments and spaces in DIN programs. All comments in curly brackets are deleted and groups of more than one space are reduced to one. This is probably recommended for reasons of saving memory space in Flash-PROM of the control. However, this setting is only suitable if the programs do not have to be viewed or changed in the control.

Value	Meaning
0	Transfer of DIN programs to Flash-PROM without changes (default).
1	Prior to saving in Flash-PROM, delete comments and reduce program.

Note!

If DIN programs are retransferred to the PC and this retransfer overwrites the original program on the PC, comments will also be deleted in this case.

4.3.4 MK_NCPROG_NICHT_INS_EEPROM

This machine constant prevents the transfer of programs to the internal Flash-PROM during the transmission of DIN programs via the monitor interface. This option must always be selected if the field of application of the control requires that new DIN programs are read in continuously via the monitor interface, which are no longer required after the machine is switched off. This is normally the standard procedure on machines for individual order processing.

Value	Meaning
0	Automatically transfer DIN programs to Flash-PROM (default).
1	Do not store DIN programs in Flash-PROM.

As the rams can only take a limited number of write cycles (approx. 100000), it is recommended in the above-mentioned case to set these MCs to 1.

4.3.5 MK_METRISCH

This machine constant selects one of the two measuring systems for all position and speed specifications. This applies to all input and output values for positions and speeds except the machine constants themselves. Irrespective of their contents, these must always be entered in the metrical system.

Value	Meaning
0	Dimensions according to MK_CONST_REL_INCH [inch] or [inch/min]
1	Dimensions according to MK_CONST_REL_MM [mm] or [mm/min] (default)

4.3.6 MK_CONST_REL_MM

This machine constant defines the input resolution in the metrical system. 1, i.e. 1 mm / unit, is used as the default. If all inputs and outputs of the control are to be specified in μm units, enter the value 0.001.

4.3.7 MK_CONST_REL_INCH

This machine constant defines the input resolution using the inch system. 25.4 is the default value, i. e. 25.4 mm / unit. If you prefer 1/10th inch units for the input and output of the control, enter 2.54 here.

4.3.8 MK_KONTURFEHLER

This machine constant defines the maximum permissible (calculated) deviation [mm] from the programmed path if the profile transition is non-tangential. Together with MK_RADIUS_B_BEWERTUNG and MK_BAHNBESCHL, it determines to which extent the speed on the path at a corner must be reduced if Look Ahead (G60) is switched on.

In the process, the control determines the maximum speed at a corner as follows:

$$V_{\max} = \sqrt{R \cdot B}$$

V_{max} max.: speed at the corner [mm/s]

R: Radius of a tangential circular arc with a distance of MK_KONTURFEHLER between the programmed reference points and the vertex of the circular arc [mm]

B: Set path acceleration (MK_BAHNBESCHL, G201) [mm/s²]

The speed is also limited by the maximum permissible jerk (increase of acceleration per time unit):

$$V_{\max} = \sqrt[3]{R^2 \cdot B \cdot K/T}$$

T: Time constant for the permissible increase of acceleration [s] MK_T_BAHNBESCHL

K: Value of MK_RADIUS_B_BEWERTUNG

In general, the following applies: the smaller the value in the MC, the smaller the resulting speed in corners and the higher the path quality. The same applies in the reverse case.

4.3.9 MK_LAH_GRENZWINKEL

This machine constant limits the Look Ahead function (G60) to a defined angle range. During this process, the system switches automatically to exact positioning on non-tangential block transitions, on which the transition angle is larger than the value of this MC.

This is to ensure that these corners in the profile can be precisely approached even when Look Ahead is switched on by forcing the speed on the corner down to 0.

The speed on the block transitions on which the transition angle is less the value of this MC, is not reduced to 0 but only as defined in MK_KONTURFEHLER.

Value	Meaning
0	If there is even a slight change of direction, this will lead to a standstill at the block transition.
15	Only a change of direction in excess of 15° will lead to a standstill.
45	Only a change of direction in excess of 45° lead to a standstill (default).
180	Special case: no speed reduction at block transitions.

Permitted values for this MC range between 0 and 180°. Please bear in mind that the special case 180° may lead to acceleration jumps at the block transitions and thus to incorrect SW limit switch messages in the look-ahead SW limit switch function (MK_SW_ENDS_MIT_RAMPE = 2).

4.3.10 MK_RADIUS_B_BEWERTUNG

This machine constant is a factor, which can be used to affect the maximum traversing speed when machining curves. The control limits the speed when machining circles depending on the circle radius and the permissible path speed per time unit (jerk) as follows:

$$V_{\max} = \sqrt[3]{R^2 \cdot B \cdot K/T}$$

$V_{max:}$ max. speed on the circular arc [mm/s]

R: radius of the circular arc [mm]

B: Set path acceleration (MK_BAHNBESCHL, G201) [mm/s^2]

T: time constant for the permissible acceleration increase (MK_T_BAHNBESCHL) [s]

K: Value of MK_RADIUS_B_BEWERTUNG

This MC is thus an evaluation factor for the permissible jerk on circular arcs. It can be used to correct the speed limit for small radii upwards or downwards.

4.3.11 MK_EPSILONMM

This machine constant is used for tolerance analysis [mm] when specifying translatory positions. It is currently taken into account during circle center point programming only, where it determines exactly how the circle's radius and center are to be programmed.

During the interpretation of a circle, the control reports a fault if the programmed radius of a circle is less than this MC or if the difference between the starting radius and the target radius is bigger than this MC. The starting radius is the distance between the programmed center and the starting point, and the end radius the distance between the target point and the programmed center of the circle.

4.3.12 MK_EPSILONGRAD

This machine constant is used for tolerance analysis [grad] when specifying rotative positions. It only affects the programming of direction-dependent modulo 360° axes (see MK_ACHSENART). This is where it defines whether the programmed traversing movement is executed or suppressed.

Example:

A rotation axis is to be positioned to +90°. The sign defines the direction. Depending on whether the axis was previously at 89.5 or 90.5, it would now have to be traversed either by half a degree or by 359.5 degrees.

MK_EPSILONGRAD intervenes by suppressing the traversing movement if the traverse path is larger than 360°-MK_EPSILONGRAD in the absolute dimension system.

4.3.13 MK_OVERRIDEMAX

This machine constant limits the override specification to this value (input in 0.1 %).

4.3.14 MK_FEHLERRESTART

This machine constant activates the function "Wiederanlauf nach Fehler" ("Restart after error"), which allows you to restart at the point of interruption or at a point previously defined in the program after a global minor error (error class 3) in the program. After an error of this kind, the program goes into standby mode which can be exited via Start (possibly Start after G10) for restart or Stop.

Value	Meaning
0	No restart possible, global minor error leads to program termination (default).
1	Restart after error possible.

See also description of G10 (📖 106).

4.3.15 MK_S0T0_VERSATZ_ERLAUBT

This machine constant is used to shift coordinate systems S0 and T0 which are intended as reference coordinate systems. The shift of these coordinate systems is normally not allowed in order to ensure that there is always one coordinate system that has not been shifted.

However, sometimes it may be useful to shift S0 or T0. In this case, this MC must be set.

Value	Meaning
0	Coordinate shift of S0 or T0 not allowed (default)
1	Coordinate shift of S0 or T0 allowed.

Please also read the descriptions on the ”S functions” (📖 178) and ”T functions” (📖 179). This is where you can find important information on the coordinate systems.

4.3.16 MK_S_VERSATZSPERRE

This machine constant protects all of the workpiece coordinate systems from the specified value against accidental shifting in DIN programs. The configured value is shown in P613 in the parameter field of the control.

Value	Meaning
0	Shift of the workpiece coordinate systems S1 ... S99 is allowed.
1 ... 99	Shift of workpiece coordinate systems Sn ... S99 is inhibited, while n is the specified value in MK_S_VERSATZSPERRE.

4.3.17 MK_CANOPEN_BAUDRATE

This machine constant defines the bit rates for the two CAN busses of the control. This MC has two parameters. The first one can be used to configure the bit rate for the IO-CAN bus (CAN1). A value of 0 at this point means that no CAN-Open modules are connected. The second parameter defines the bit rate for the Drive-CAN bus (CAN2).

Value	Meaning
0	No CAN-Open modules connected.
125	Bit rate 125 kBit/s
250	Bit rate 250 kBit/s
500	Bit rate 500 kBit/s
1000	Bit rate 1000 kBit/s

4.3.18 MK_DELTAT

This machine constant sets the internal interpolation cycle [ms] of the control. During each of these rough interpolation cycles, the control calculates new position values for the participating axes from the programmed path. The smaller the cycle, the closer are the calculated positions to one another. However, this also means that the calculation load for the control and the bus load for the drive amplifier with digital interface is higher. The ETC can operate with a minimum rough interpolation cycle of 1 ms.

If the amplifiers are provided with a CAN interface, the limited bandwidth of the CAN bus transfer is extremely important. In the following table, you can find the minimum rough interpolation cycle and the average CAN2 bus load depending on the number of configured CAN axes and the set bit rate.

For the calculation of the bus load, only the cyclic telegrams were taken into account, while any additional asynchronous telegrams from ETC-MMI or PLC were not. The specified value may vary slightly due to different telegram lengths for different drive amplifiers.

Number of axes	1000 kBit	500 kBit	250 kBit
1	1 ms / 29 %	2 ms / 29 %	3 ms / 38 %
2	1 ms / 52 %	2 ms / 52 %	4 ms / 52 %
3	2 ms / 38 %	3 ms / 50 %	5 ms / 60 %
4	2 ms / 49 %	3 ms / 66 %	6 ms / 66 %
5	2 ms / 61 %	4 ms / 61 %	7 ms / 70 %
6	2 ms / 73 %	4 ms / 73 %	8 ms / 73 %
7	3 ms / 56 %	5 ms / 67 %	9 ms / 75 %
8	3 ms / 64 %	5 ms / 77 %	10 ms / 77 %
9	3 ms / 72 %	6 ms / 72 %	11 ms / 78 %
10	3 ms / 79 %	6 ms / 79 %	12 ms / 79 %
11	4 ms / 65 %	7 ms / 75 %	13 ms / 80 %
12	4 ms / 71 %	7 ms / 81 %	14 ms / 82 %

Once it has been established that no asynchronous telegrams are transferred by MMI or PLC during operation, the use of the available transfer bandwidth can be optimized, see the following table (no guarantee):

Number of axes	1000 kBit	500 kBit	250 kBit
1	1 ms / 29 %	1 ms / 57 %	2 ms / 57 %
2	1 ms / 52 %	2 ms / 52 %	3 ms / 69 %
3	1 ms / 75 %	2 ms / 75 %	4 ms / 75 %
4	2 ms / 49 %	3 ms / 66 %	5 ms / 79 %
5	2 ms / 61 %	3 ms / 81 %	6 ms / 81 %
6	2 ms / 73 %	4 ms / 73 %	7 ms / 83 %
7	2 ms / 84 %	4 ms / 84 %	8 ms / 84 %
8	2 ms / 96 %	4 ms / 96 %	8 ms / 96 %
9	3 ms / 72 %	5 ms / 86 %	9 ms / 96 %
10	3 ms / 79 %	5 ms / 95 %	10 ms / 95 %
11	3 ms / 87 %	6 ms / 87 %	11 ms / 95 %
12	3 ms / 95 %	6 ms / 95 %	12 ms / 95 %

4.4 Storage space reservation

The following MCs affect the static memory layout within the control. The set values only become effective after they have been changed on the control or transferred from the PC and the control has been restarted.

4.4.1 MK_SPS_SPEICHERGROESSE

This machine constant reserves the program memory for the PLC in the internal main memory of the control and thus defines the maximum size of the PLC program to be loaded in kByte at the same time. The control reserves twice the memory specified in the MC in order to support the ”online change” function of the PLC programming interface. The memory reserved in this manner is thus permanently assigned to the PLC and is no longer available for other tasks of the control.

The default value of 128 kByte is sufficient for smaller PLC programs. In the case of larger programs, this value must be increased accordingly.

4.4.2 MK_SPS_DATENGROESSE

This machine constant defines the size of the data memory for the variables of the PLC in kByte. This memory is permanently reserved for the PLC and is no longer available for other tasks in the control.

The default value of 32 kByte should be sufficient for most PLC programs.

4.4.3 MK_SPV_SPEICHERGROESSE

This machine constant defines the size of the internal main memory in RAM for managing DIN programs (also online programs). The specification is the number of storage blocks at 1024 byte (1 kByte). Their value can be increased or decreased. Reasonable values start from 64 K. The maximum value at 4 MByte memory capacity is 2048 kByte.

Note!

The memory space required by the DIN programs in the internal main memory is approx. 2 ... 4 times as much as the size of the corresponding ASCII file. This is due to the fact that the programs are already preinterpreted and stored in an intermediate code in the main memory for a faster execution during loading.

4.4.4 MK_SPV_SYMBOLANZAHL

This machine constant defines the size of the symbol table, which is required for managing symbolic program numbers. If a larger value than 0 is entered for this MC, symbolic program names can be used when programming the DIN programs. During this process, dynamic program numbers between 32768 ... 65534 were assigned program names. For further information on this, refer to chapter ”CNC programming”.

4.4.5 MK_HEADERANZAHL

This machine constant determines the number of ”head entries” for the management of the DIN programs in the main memory. An entry of this kind is required in each DIN program in the main memory. Each head entry takes up 32 byte in the internal main memory (see MK_SPV_SPEICHERGROESSE).

To store a max. of 100 DIN programs in the main memory at the same time, use a default value of 100.

4.4.6 MK_KANALANZAHL

This machine constant determines the number of NC channels, in which DIN programs can be executed at the same time.

Value	Meaning
1	Only one channel is provided for the execution of DIN programs (default).
2	Two channels are provided for the simultaneous execution of DIN programs.

Note!

For each channel, additional resources of calculating time and main memory are required. Therefore, only increase this MC if you really need several NC channels.

Please bear in mind that a program in the second channel cannot be started via the start key. It can only be started from a program in the first channel. For further information on multi-channels, refer to chapter ”CNC programming” in the descriptions for G22 (🕮 109) and G153 (🕮 150).

4.4.7 MK_LAH_VORLAUFTIEFE

This machine constant defines the maximum number of orders in the prebuffer. The prebuffer is a circular puffer (FIFO) between DIN interpreter and rough interpolator and is used to isolate the interpretation from the execution of DIN blocks. This MC defines how many blocks can be seen in advance during the processing of a DIN program.

The MC contains one parameter for each possible NC channel.

The minimum value of this MC is 8 blocks. The entered value should be a power of two (8, 16, 32, 64, 128, ...). If not, it is internally rounded up to the next higher power of two.

The prebuffer is assigned resources in the main memory, namely approx. 1500 byte per block. If the size of the memory is insufficient for installing the prebuffer, a corresponding error message is generated when restarting the control and the prebuffer is reduced to the minimum size of 8 blocks.

Please also read chapter ”Block preprocessing and time synchronization” in chapter ”CNC programming” (📖 90).

4.4.8 MK_LAH_RUECKLAUFGRENZE

This machine constant defines the maximum number of blocks which can be interpolated backwards in a program. These blocks are stored in the prebuffer and reduce the predepth, see above.

The MC contains one parameter for each possible NC channel.

The value in this MC must not exceed MK_LAH_VORLAUFTIEFE-8. Otherwise, the return limit is limited to this value and an error is reported.

Note!

The number of blocks to be interpolated backwards is also limited by other factors. In general, the backward interpolation stops when no more interpolation blocks are available. It stops as soon as a function occurs which is not a pure interpreter function and cannot be inserted in the previous or subsequent block, e.g. an individual M function, which is not preceded by a traverse block or a traversing block with M or Q function. (📖 174).

Please make sure that M functions are output by default during backward interpolation, but no Q functions.

4.4.9 MK_PFELD_GROESSE

This machine constant defines the total size of the parameter field in the control, including the 1024 system parameters. The MC specifies the number of parameters in the parameter field. Each parameter is assigned 8 byte of the main memory. The minimum size is 2048 parameters.

For further information on the parameter field, refer to chapter ”P-field” (📖 180).

4.5 Configuration of axes - Basics

These are the most important settings that have to be implemented as they are used to adapt the control to the mechanics.

As the control can be used for the many different axis combinations, the following must be specified by means of the machine constants:

- ► which type of axes exist on this specific machine,
- ► with which identification letter the axes are to be addressed,
- ► to which (hardware) axis computer channel the corresponding (software) axis is connected.

In the control, the axes are consecutively numbered from 0 ... N-1, while N is the maximum number of configurable application (=software) axes. This number is control-specific and does not have to correspond to the maximum number of the physical (=hardware) axes. At the end of this chapter, you can find a table in which the maximum number of axes is listed for the individual control types.

After it has been established which physical axes exist on the machine, the application axes must be assigned to the physical axes.

4.5.1 MK_CANDRIVES

MK_CANDRIVES defines the axes that are connected to the second CAN bus.

Enter a valid (0 ... N-1) application axis number or the value -1 in each parameter of this MC with N as the maximum number of application axes. If two physical axes are assigned to one application axis, which means that one application axis number appears twice in one of these MCs, this is called a synchronous axis. This is the only case, when the number of configured physical axes differs from the number of configured application axes.

After it has been established which application axes exist, letters must be assigned to these axes. The axes in the DIN program can only be programmed via the axis letters.

4.5.2 MK_APPLACHSIDX

This MC has 18 parameters, one for each of the 18 possible axis letters. The order of the letters assigned to the parameters is fixed. Only the assignment of the application axis numbers can be freely selected. The order of the letters is as follows

X Y Z C U V W A B u v w a b c x y z

Exactly one application axis can be assigned to each letter. For the letters that are not used, enter -1 or any non-configured application axis number.

The control only searches the axes that are configured in MK_CANDRIVES and assigns the first axis letters it finds to the axes. Therefore, if you specify one application axis number more than once in this MC, the first letter is used and the others are ignored.

The assignment of axis letter is completely optional. However, you should observe the following rules which make it easier for others to understand your machine.

Identify the linear principle axes of the machine with X, Y, Z and the rotative principle axes with A, B, C. Define X, Y, Z in such a way that you specify a clockwise workpiece coordinate system and that the rotation axis A rotates around the X, rotation axis B around the Y and C around the Z axis. The axes U, V, W should be used for linear auxiliary axes with U moving parallel to X, V parallel to Y and W parallel to Z.

In MK_APPLACHSIDX the application-related order of axis letters is defined by means of numerical assignment. This order of axes is valid for all axis-related machine constants!

Example

Your task is to configure a machine that has an X, Z and C axis. The X axis must be a synchronous axis.

The MK_CANDRIVES assignment must be as follows:

MK_CANDRIVES	0,	/* X axis */
	0,	/* X' axis */
	1,	/* Z axis */
	2,	/* C axis */
	-1,	/* not assigned */
	-1,	
	-1,	
	-1;	
MK_APPLACHSIDX	0,	/* X */
	-1,	/* Y */
	1,	/* Z */
	2,	/* C */
	-1,	/* U */
	-1,	/* V */
	-1,	/* W */
	-1,	/* A */
	-1,	/* B */
	-1,	/* u */
	-1,	/* v */
	-1,	/* w */
	-1,	/* a */
	-1,	/* b */
	-1,	/* c */
	-1,	/* x */
	-1,	/* y */
	-1;	/* z */

In this case, value -1 serves to specify that this axis is not configured. The X axis is defined as the first, the Z axis as the second and the C axis as the third application axis. I.e. for all following MCs of this category, the first parameter is assigned to the "X" axis, the second parameter to the "Z" and the third parameter to the "C" axis.

The number of configurable axes depends on the stage of the control.

4.6 Configuration of axes - Assignment and evaluation

4.6.1 MK_CANDRIVES

This machine constant assigns application axis numbers to the node numbers 1 ... 12 on the second CAN bus and thus defines which application axes are configured. For each unassigned node number, enter value -1. The axis letters are assigned to the specified application axis numbers via MK_APPLACHSIDX.

The MC has a maximum of 12 parameters for the 12 possible node numbers on the bus. The node number assignment to the parameters is consecutively numbered in a fixed order from 1 ... 12.

If an index is entered twice within MK_CANDRIVES, a forced coupling is generated between two axes (synchronous axes). For further information on this, refer to chapter ”Synchronous axes” (📖 220).

4.6.2 MK_APPLACHSIDX

This machine constant assigns letters to the application axis numbers configured with MK_HARDKONF, MK_ESABKONF and MK_CANDRIVES and thus defines which application axis letters are accepted by the control.

This MC has 18 parameters, one for each of the 18 possible axis letters. The order of the letters assigned to the parameters is fixed. Only the assignment of the application axis numbers can be freely selected. The order of the letters is as follows

X Y Z C U V W A B u v w a b c x y z

Please make sure that the sequence of the parameters of all following axis-specific MCs corresponds to the application axis numbers, i.e. 1st parameter = axis number 0, 2nd parameter = axis number 1, nth parameter = axis number n-1. The unconfigured application axis numbers are included in the count.

This way, this MC defines in which order the parameters of the axis specific MCs are to be specified in relation to the axis letters.

Please also observe the specifications in the previous chapter!

4.6.3 MK_ACHSENART

This machine constant specifies different axis properties. The MC is bit coded.

Bit	Value	Meaning
0	0 1	linear axis rotation axis
1	0 2	observe limit switch ignore limit switch
2 ... 3	0 4 8 12	normal axis spindle measurement axis spindle + measurement axis
4... 5	0 16 48	normal axis modulo 360° axis, the sign determines the traversing direction modulo 360° axis, the angle position is approached via the shortest path [1)]
6	0 64	normal axis gantry axis, only effective in the case of synchronous axes
7	0 128	normal axis handwheel

To set the desired bit, add the value in the table to the value of the MC.

Example

For a rotation axis, which is to ignore the limit switch signals, and if modulo 360° is programmed with the sign defining the traversing direction, you would have to enter value 19 (bit 0, 1 and 4).

Please bear in mind that handwheels are consecutively numbered within the control. Therefore, if you declare several handwheels, the first one is (the one with the smallest application number) is identified as handwheel 0, the next one as handwheel 1, etc...

A gantry axis is a special form of a synchronous axis. It includes a mechanical coupling between master and slave axis. In contrast to simple synchronous axes, the home position approach for gantry axes, which have an analog interface, is carried out synchronously.

4.7 Configuration of axes - Resolution

4.7.1 MK_IMPULSE

This machine constant determines the number of pulses assigned to the actual value counter of the axis interface for rotary motors per revolution on the motor shaft and for linear induction motors per millimeter. Enter the number of pulses including the pulse quadruplication; e.g. for ECS compact servo 65536 imp. / revolution.

By specifying a negative value, the definition direction of the axis is reversed, i.e. it moves into the opposite direction. Changing the sign when specifying MK_WEG has the same effect.

The same applies to the home position approach (must be taken into consideration for MK_REF_RICHTUNG_UND_FOLGE)!

4.7.2 MK_WEG

This machine constant determines the distance at which the axis computer counts the increments entered under MK_IMPULSE, i.e. in the case of rotary motors, the distance per revolution on the motor shaft, and in the case of linear motors, 1 mm. Millimeters are used as a unit for linear axes, degrees for rotation axes.

Example

MCS06 servo motor with ECS compact servo (65536 pulses). The motor goes via a gearbox (4:1) to a spindle, which moves a slide backwards by 3.4 mm per revolution. I.e. if a positive distance is specified, the mechanics move in negative direction.

```
MK_IMPULSE  65536,...;
MK_WEG     -0.85,...;   /* 3.4 / 4 */
```

4.7.3 MK_MASSSTAB

This machine constant defines a multiplication factor for axis-specific position specifications and position indicators. This is useful if diameters are to be programmed in the case of round workpieces, but the feed speed is to be based on the radius at the same time.

A scale factor for X of 0.5 means that a position specification of 10 mm results in a traversing movement of X by 5 mm. However, the display only shows the 10 mm.

4.8 Configuration of axes - Operating range

4.8.1 MK_GRUNDOFFSET

This machine constant is the offset of the mechanical zero point of the machine in relation to the zero point of the position measurement system. The unit is millimeters for linear axes and degrees for rotation axes.

4.8.2 MK_SW_ENDS_MINUS, MK_SW_ENDS_PLUS

This machine constants define the positive and negative traversing range limit of the mechanics. The millimeter unit is used for linear axes and degrees for rotation axes. Use G26 to switch on the monitoring function. See also MK_VOREINSTELLUNG.

The values refer to the machine zero point (home position + basic offset) and not to the zero point of the position measurement system.

4.8.3 MK_SW_ENDS_MIT_RAMPE

This machine constant indicates whether a controlled deceleration or an abrupt stop should take place when the traversing range limits are exceeded.

Value	Meaning
0	When the SW limit switches are reached, there is an abrupt stop.
1	Decelerate on the programmed path with the current deceleration ramp. In the process, set the traversing range limit in such a way that the axis can stop before reaching the mechanical limit stop by means of the set acceleration values.
2	Look-ahead software limit switches. The process involves the calculation of whether the corresponding software limit switch will be reached by means of the current axis speed and the configured max. axis deceleration ramp. If this is the case the axis in question is decelerated to such an extent that it comes to a halt at the set software limit. All of the other axes which participate in the path are stopped as in the case of 1. Caution: The correct functioning of this function can only be guaranteed if a value less than 180° is entered in MK_LAH_GRENZWINKEL.

4.8.4 MK_SCHLEPPGENAUHALT

This machine constant defines the limit value of the system deviation of the individual axes for exact positioning with following error monitoring in [mm] or [degrees]. This function must be activated with G60.(🕮 118).

4.8.5 MK_GENAUHALTZEIT

This machine constant defines the time in seconds by which the extent of the following error of an axis must have constantly fallen short of the configured value under MK_SCHLEPPGENAUHALT, before the next function can be executed. This MC is only effective if the ”Genauhalt mit Schleppabstandsüberwachung” (”Exact positioning with following error monitoring”) function is switched on. The default value is 0.

4.9 Configuration of axes - Controller settings

4.9.1 MK_T2

This machine constant is the filter time constant for the fine interpolation filter in seconds. This MC does not affect the position control directly, but the fine interpolation by means of which new position setpoints are calculated in the fine interpolation grid (MK_FIT_PRO_GIT). The fine interpolation filter is a filter of the 2nd order which can be used to smooth the calculated position setpoints to ensure that specific axes move more smoothly. This MC also affects axes with a digital interface.

The fine interpolation filter causes a time delay with regard to the output of the position setpoints. This delay is approx. twice as long as the filter time constant. Therefore, the same filter time constant should always be entered for axes which are included in one path interpolation.

This MC is set to 0 by default and should only be increased if an axis tends to oscillate in the case of discontinuities on the path, e.g. at transitions from one radius to another radius. This affects mainly axes that have a great reverse backlash and toothed racks. Reasonable values range between 0 and 0.2 s. The higher the set value, the greater the path deviation which is caused by the filter.

4.10 Configuration of axes - Referencing

4.10.1 MK_REF_RICHTUNG_UND_FOLGE

This machine constant determines the direction and the sequence, in which the axes carry out a home position approach. The direction, into which the axis is to move first, is defined by the sign of the entered value. The sequence, in which the axes carry out the home position approach during joint referencing, is defined by the amount of the value.

If an axis should not participate in an automatic home position approach, enter 0 or -128 for this axis. -128 indicates an inversion of the direction during the manual home position approach of the axis.

Example

The machine has an X, Y, Z and a C axis. Make sure that the Z axis has traveled out of the working area before X and Y (MK_APPLACHSIDX with default values) are moved. The Y axis must search the cam in negative direction. The C axis must not be included in the automatic home position approach.

MK_REF_RICHTUNG_UND_FOLGE 2, -2, 1, 0;

4.11 Configuration of axes - speed and acceleration

There are different MCs which influence the maximum speeds and ramps for the individual axes and for the paths resulting thereof. These are the limit values that must not be exceeded by the control. The axis-specific MCs are effective in path, route and manual operation. The path-specific MCs only act as limit values in path operation.

4.11.1 MK_MODVMAX

This machine constant is the maximum speed for manual operation (traverse buttons) of the individual axes. The specified value is limited to MK_VMAX.

4.11.2 MK_VMAX

This machine constant is the maximum permissible speed of the individual axes in [m/min] or [1/min]. This corresponds the high-rate speed of the axes in route operation. Also the traversing speed in path operation is limited by the control, if possible, to such an extent that the maximum speed of the individual axes is not exceeded.

Note!

In the case of some transformations, such as basic/profile rotation and 6-axes transformation, the control cannot limit the material speed because, seen logically, the transformation takes place after the path interpolation and the die speed of the axes cannot be determined beforehand.

4.11.3 MK_BESCHL, MK_BREMS

These machine constants define the maximum acceleration and deceleration ramps of the individual axes in [m/s^2] or [$1/s^2$]. During path operation, the path ramps are limited in proportion in accordance with the involvement of the individual axes.

4.11.4 MK_T_BESCHL

This machine constant is the time in milliseconds, in which the maximum ramps configured in MK_BESCHL and MK_BREMS are to be reached in manual operation and route operation. This MC is used to set acceleration ramps and deceleration ramps of the individual axes that are similar to $\sin^2$ ramps.

4.11.5 MK_VBAHNMAX

This machine constant defines the maximum speed [m/min] in path operation. It may be higher than the maximum speed of the individual axes if the resulting speed of the participating axes is less or equal to MK_VMAX.

4.11.6 MK_BAHNBESCHL, MK_BAHNBREMS

These machine constants define the maximum permissible acceleration and deceleration ramps [m/s^2] in path operation. Depending on the axes included, the path ramps are additionally limited such that the resulting ramps of the individual axes do not exceed the values of MK_BESCHL and MK_BREMS.

4.11.7 MK_T_BAHNBESCHL

This machine constant is the time in milliseconds, in which the ramps specified under MK_BAHNBESCHL and MK_BAHNBREMS are to be reached. This MC is used for setting acceleration and deceleration ramps on the path which are similar to $\sin^2$.

4.12 Configuration of axes - Correction of axes

4.12.1 MK_SPINDELUMKEHRSPIEL

This machine constant uses millimeters or degrees to define by which value the set position is to be corrected if the travel direction is reversed. The default value is 0. As soon as a value is entered, the spindle reverse compensation is switched on.

Correction of a spindle gradient distortion

The correction of a spindle gradient distortion or linearity distortion of the position measurement system can be carried out by means of a table which is stored in Flash-PROM or transferred by ETC-MMI after the control has been started up. The correction is switched on automatically after the table has been transferred.

The correction is carried out for the axes specified in the file within the defined actual positions. The positions refer to the home position. In the case of a periodically repeated error, it is possible to define the relative range by means of specifying a ”modulo” value within the defined correction area. Once the end of this relative area is reached, the values from the top of the table are processed again.

For the correction values one byte each is available, i.e. the correction value can be -128 ... 127 increments per position value. During the axis output, the correction value is added to the calculated position. A linear interpolation takes place between two grid points. Positions that lie outside the defined range are not subject to correction. The correction table can contain a maximum 4096 entries per axis.

Structure of correction file

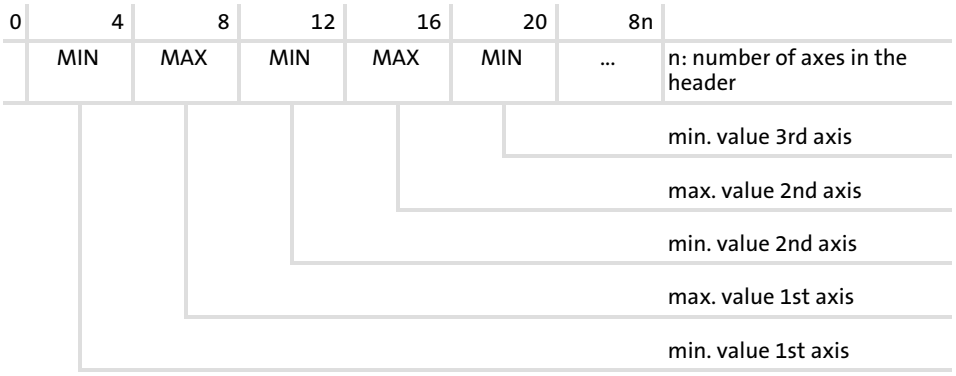
The file is a binary file. It must be stored under the name ”ACHSEN.KOR” in Flash-PROM (load with utility program ”download”). The file consists of a header with a length of 128 byte, 2 definition blocks and up to 512 data blocks of 8 byte each per axis. The data is stored in binary format.

128	8n	8n	8n	8n
Header	Def 1: MIN/MAX	Def 2: MOD	Data (1)	Data (512)

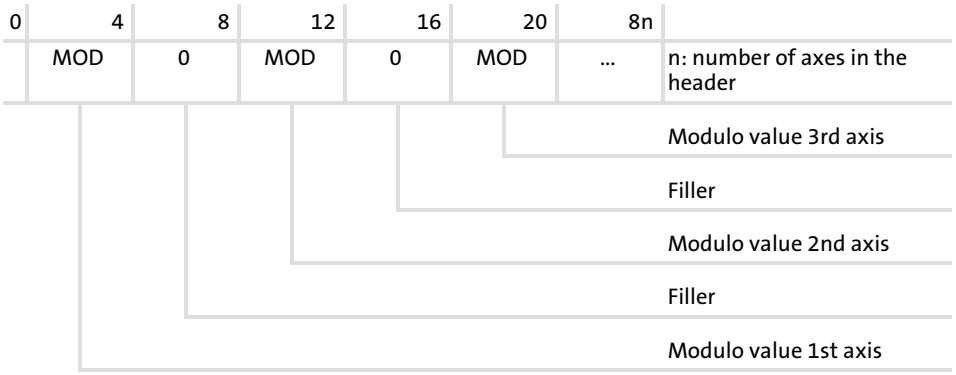
Header: The header contains the null-terminated list of the axis letters of the participating axes. They can be in any order. The data of the following blocks are interpreted in this order. The rest must be filled with binary zeros.

0	1	2	3	n	127	128
	X	Y	Z	...	0	...	0

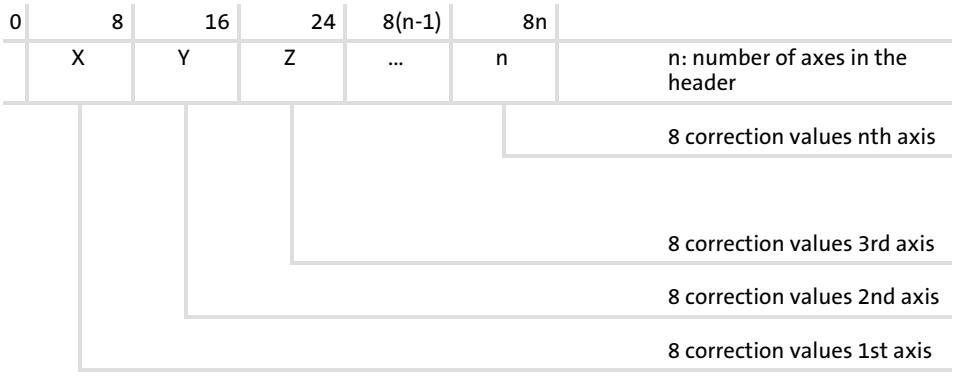
Def 1: The first definition block contains per axis in the header one 32-bit integer for minimum and maximum value of the actual position range, in which the correction table is valid. The specification is made in increments in relation to the home position.



Def 2: The second definition block contains per axis one 32-bit integer for the modulo value. The modulo value defines the relative actual position range covered by the correction table. This range can be repeated multiple times between the minimum and maximum value. The specification is made in increments. The remaining 4 byte per axis are not assigned and must be initialized with 0.



Data: Each data block has 8 byte of data per axis and contains the actual correction data. Each correction value is represented by one byte. Thus, on data block contains 8 correction values per axis. Each correction value has a value range of -128 ... +127 increments.



4.13 Configuration of axes - Handwheels

The control allows for the connection of electronic handwheels with CAN interface. For this purpose, the handwheel must be configured in the machine constants. Configure the handwheel analog to a normal axis.

The configured handwheels are consecutively numbered in the control from 0 ... N-1, while N corresponds to the maximum number of configurable handwheels. Therefore, the assignment of an axis letter is optional and generally not required.

The following machine constants must be set:

4.13.1 MK_CANDRIVES

This machine constant assigns application axis numbers to the node numbers 1 ... 12 on the second CAN bus and thus defines which application axes are configured. For each unassigned node number, enter value -1. The axis letters are assigned to the specified application axis numbers via MK_APPLACHSIDX.

4.13.2 MK_APPLACHSIDX

Assignment of a free axis letter for the configured handwheel. The assignment of a letter is optional and only required if the handwheel or the configured application axis channel is to be addressed via other G functions as G187, e.g. home position approach with G74.

4.13.3 MK_ACHSENART

For the application axis assigned by MK_CANDRIVES, bit 7 must be set or value 128 entered.

4.13.4 MK_IMPULSE

Number of pulses, which the handwheel provides per defined distance.

4.13.5 MK_WEG

Distance which an assigned axis should cover per number of pulses (MK_IMPULSE) at an evaluation factor of 1.0. The direction can be changed by changing the sign.

4.13.6 MK_HANDRADZUORDNUNG

This machine constant is an axis-specific MC which assigns a configured handwheel to the individual axes. This MC has a parameter for each application axis in which the number of the assigned handwheel is entered. It must only be changed if more than one handwheel is configured. The assignment can only be changed via this MC.

4.13.7 MK_HANDRADFAKTOR

This machine constant is an additional axis-specific evaluation factor for the handwheel function. It is used to obtain different evaluation factors for the individual axes, e.g. when linear as well as rotation axes are operated on one handwheel.

4.13.8 MK_HANDRADFILTER

This machine constant is a filter time constant [ms] for the handwheel function. It can be used to smooth the speed specification of the handwheel to make the traversing movement softer via the handwheel. This is normally only necessary for handwheel evaluation factors and low handwheel resolution, e.g. evaluation factor 100 and one increment per grid on the handwheel.

Value	Meaning
0	Handwheel filter switched off (default)
100	100 ms filter time constant, corresponds to a limit frequency of 10 Hz
250	250 ms filter time constant, corresponds to a limit frequency of 4 Hz

The value can be set to any value between 0 and 500 ms.

The handwheel filter causes a time delay of the position specifications from the handwheel to the selected axis by the set value. Therefore, the time constant should not be set to a value which is too high. Otherwise, the response times are extended unnecessarily.

4.14 Configuration of axes - Synchronous axes

A synchronous axis is configured in the control by entering the corresponding axis number in two places in MK_CANDRIVES. This way, a forced coupling is generated between the two physical axes.

The axis channel with the lower index is automatically the master axis. The slave axis channel, which has been assigned the same axis identification, is the slave axis. The slave axis executes all entries for the master axis synchronously.

If a forced coupling is configured, the following machine constants must also be specified.

4.14.1 MK_ACHSENART

In this MC, Bit 6 must be set if the synchronous axis is a gantry axis. If this is the case, there is a mechanical coupling between the master and the slave axis. For axes with an analog interface this means that the home position approach is already carried out synchronously. In the case of axes with a digital interface, the home position approach is merely started synchronously.

4.14.2 MK_SYNCHRONABWEICHUNG

This machine constant defines the maximum permissible deviation between master and slave axes position for synchronous error monitoring. Depending on the axis type, the unit is either millimeters or degrees. This machine constant is only important if a forced coupling was configured.

4.14.3 MK_SYNCHRONOFFSET

This machine constant defines the distance between the home positions of master and slave axis in [mm] or [degrees]. It should ideally be 0 in order to ensure that both axes are already synchronous after the home position has been reached. The master axis continues to travel by the value which is entered here after reaching the home position, while the master axis stops at the home position.

4.15 Technology-specific settings

4.15.1 MK_MFKT_UPR_TABELLE

This machine constant is a list of up to 16 M function numbers, which generate a branch in a subprogram during processing within a DIN program. A difference is made between the following cases:

Value	Meaning
< 500	When the configured M function appears in the program, a subprogram call with the program number 9000 + M function number is inserted subsequently. The programmed M function is output before the first executable block of the subprogram is executed.
500 600 700 800 900	Each of these values stands for the following 99 M function numbers. Every time one of these 100 M functions appears in the program, the M function is replaced with the program number 9000 + basic M function number by means of a subprogram call. Therefore, M621 becomes G22 L9600. The programmed M function is not output. Therefore, no processing time is required in the PLC for the M function.

The M function, which lead to the subprogram call, is stored in the parameter field in P562 for NC channel 0 and in P594 for channel N. It can be evaluated in the subprogram.

For further information, refer to chapter ”M functions” (📖 175).

4.15.2 MK_TECHNOLOGIEDATEN1 ... MK_TECHNOLOGIEDATEN4

These machine constants contain 10 entries each, which are stored one-to-one in the parameter field of P760 ... P799 where they can be used as required. For the assignment of the parameter field, refer to chapter ”P field” (📖 180).

4.15.3 MK_MASCH_POLAR_KART

This machine constant switches from a Cartesian to a polar machine coordinate system. In the polar coordinate system, the positions in the machine plane are defined by an angle axis and a radius axis, while the control transfers the transformation of the interpolated Cartesian coordinates into the polar coordinate system.

During this process, programming and interpolation remain in Cartesian coordinates both during manual traversing and in program operation. The Cartesian and polar axes are assigned to the coordinate systems via MK_KARTESISCH_ACHSNR and MK_POLAR_ACHSNR.

Value	Meaning
0	Cartesian machine coordinate system (default)
1	polar machine coordinate system

Note!

The speed within the interpolator is only limited in the Cartesian coordinate system. The maximum speeds of the polar radius axis and angle axis can no longer be monitored after the transformation. In an extreme case, e.g. traveling through the polar zero point, this may lead to following errors.

4.15.4 MK_KARTESISCH_ACHSNR

This machine constant defines the numbers of the Cartesian axes in the case of a polar machine coordinate system. The positions of these virtual axes are used to calculate the positions of the polar radius and angle axis later on. The numbers of the axes should be the same as the polar axes to ensure that only the number of axes that are physically available must be configured.

This MC has the following three entries:

Index	Default	Meaning
0	0	Number of the cosine axis (X)
1	1	Number of the sine axis (Y)
2	-1	Number of the optional tangential correction axis or -1 (C)

The first two entered axes must be configured as linear axes in MK_ACHSENART. All unit dependent MCs, except for MK_WEG, must be specified in [mm] for these axes. The third axis must be configured as a rotation axis.

4.15.5 MK_POLAR_ACHSNR

This machine constant defines the numbers of the radius and the angle axis in the polar machine coordinate system. They are the physical axes for radius and angle on the machine.

This MC has the following 2 entries:

Index	Default	Meaning
0	0	Number of radius axis
1	1	Number of angle axis

As these should be the same axis numbers as the first two axes in MK_KARTESISCH_ACHSNR, the remaining configuration is the same. The only difference is that in MK_WEG, degrees must be used as the unit for the angle axis as it is a real rotation axis.

4.15.6 MK_WLK_C_GRENZWINKEL

This machine constant is used by different correction modules (e.g. ”SCHNEIDEN” [”CUT”]) for the tangential correction of the C axis and defines up to which relative angle an abrupt positioning of this axis may take place.

If the angles are greater than this MC, the feed on the path is stopped, the C axis is fed individually with G0 and then the path is interpolated further. Depending on the correction module, an M15 is inserted before the G0 and an M14 behind the G0 in order to carry out the positioning with the tool raised.

In the case of smaller angles the positioning of the C axis to the new profile angle takes place at the block transition in one step without affecting the path interpolation.

Value	Meaning
45	abrupt positioning up to 45° (default)
15	abrupt positioning up to 15°
90	abrupt positioning up to 90°

Any angle between 0 and 180° can be set.

Note!

Which correction module is switched on for G41/G42 depends on the technology used and the value in MK_KUNDE (📖 193).

4.15.7 MK_WLK_C_OFFSET

This machine constant is used in the correction module ”Schneiden” (”Cut”) for compensating the technology-specific pitch of the tool. The MC specifies the offset in degrees by which the C axis is to be shifted in relation to the calculated path orientation. The default value of 0 does not cause a shift.

4.15.8 MK_WLK_VERWEILZEIT

This machine constant is used to automatically insert dwell times in the correction module ”SCHNEIDEN” (”CUT”) at all places where the cutting tool is to punch and cut. The MC has two parameters. The first entry defines the dwell time after punching (M15/M16) and the second one the dwell time after cutting. Both are entered in seconds with decimal positions.

4.15.9 MK_X_WINKEL

This machine constant defines an oblique coordinate system in the Z-X plane. This is necessary, for example, for cylindrical grinding machines with an oblique grinding wheel. This MC specifies to which extent the angle of the X axis deviates from the perpendicular.

The display and in the parameter field always shows the tool tip in relation to the tool in rectangular coordinates. With regard to X axis travel, a difference must be made between path (G31) and route operation (G30). While only X moves on the display in path operation, X and Z move on the machine. The tool tip stops in relation to the tool in Z direction.

In route operation, the situation is the exact opposite. While only the X axis moves on the machine, X and Z move on the display. The tool tip moves in Z direction in relation to the workpiece.

Note!

When the X axis travels in route operation, the modal offset of the Z axis (P160ff) is shifted. This corresponds an actual shift of the NC coordinate system. If required, this shift can be undone by means of a G121 and set it against the NC actual position (P0ff).

Value	Meaning
0	Oblique X axis switched off (default)
15	X axis is offset by 15° in relation to the perpendicular
-30	X axis is offset by -30° in relation to the perpendicular

The specified angle may range between -90° and 90°.

4.15.10 MK_GEWINDE_VMAX

This machine constant limits the feed speed [mm/min] of the coupling axis during thread grinding. For further information on the thread grinding function, refer to ”ETC Programmierung NC Betriebssystem” (”ETC programming NC operating system”) for G33.

4.15.11 MK_DW224_255

This machine constant is used to affect the sequence program of the PLC. The contents of the 32 entries of this MC are copied in the data area %MW2.224 to %MW2.255 of the PLC during the transfer of the machine constants. The programmer can freely define the meaning of the data words.

This MC can be used, for example, to configure different pieces of equipment of a machine with additional components. To create one PLC version only, the equipment can be recorded in the MCs and the PLC program can react accordingly.

4.16 List of machine constants

In the following, you can find the list of machine constants as it is loaded in the control. The list is provided as an ASCII file and can be edited on a PC. The structure of the list corresponds to the previous description. It is recommended to enter the required values for all machine constants.

Do not change the key values listed. Only change or add the corresponding numerical values.

The individual numerical values within a keyword end with a ”,” (comma). The last numerical value must end with a ”;” (semicolon).

During editing, make sure not to delete the comment characters (”/*”, ”*/”) accidentally.

```
/*================================================================*/
/*         Sample machine constants V1.74 for ETCxC           */
/*----------------------------------------------------------------*/
/*
** PLEASE NOTE
** --------------
**
** Sequence of axes:
** The internal order of axes is defined via MK_APPLACHSIDX.
** All axis-specific machine constants must be configured
** in this order.
**
** Machine configuration:
** The machine constant MK_DW224_255 at the end of this file
** contains important information for the PLC. For further information,
** refer to the comment next to the machine constant.
**
*/
/*================================================================*/

/*---------------------------------*/
/*   0. Test settings           */
/*---------------------------------*/
MK_TEST_OHNEMECHANIK          0;       /* Axis computer is in operation, but does not observe the
                                       */
                                       /* Actual position counter */
MK_SPS_DUMMY                  0;       /* 0 = with integrated PLC; 1 = without */
```

```
/*-----------------------------------------*/
/*       1. Hardware configuration         */
/*-----------------------------------------*/
MK_HARDKONF                   -1,      /* Assignment axis computer outputs */
                              -1,      /* (0-7) to axes; -1 = axis not connected */
                              -1,
                              -1,
                              -1,
                              -1,
                              -1,
                              -1;

MK_CANDRIVES                   0,      /* Assignment of axes - nodes 1-12 */
                               1,
                              -1,
                              -1,
                              -1,
                              -1,
                              -1,
                              -1;

MK_ESABKONF                   -1,      /* Assignment of the axes to the axis addresses 1-8 */
                                       /* on ESA bus */
                              -1,
                              -1,
                              -1,
                              -1,
                              -1,
                              -1,
                              -1;

MK_APPLACHSIDX                 0,      /* Internal number of the X axis, */
                               1,      /* Y axis */
                               2,      /* Z axis */
                               3,      /* C axis */
                               4,      /* U axis */
                               5,      /* V axis */
                               6,      /* W axis */
                               7,      /* A axis */
                               8,      /* B axis */
                               9,      /* u axis */
                              10,      /* v axis */
                              11,      /* w axis */
                              12,      /* x axis */
                              13,      /* y axis */
                              14,      /* z axis */
                              15,      /* a axis */
                              16,      /* b axis */
                              17;      /* c axis */
MK_APPLSPINDELIDX             -1,      /* Assign. spindles - axis channels */
                              -1,      /* -1 = spindle not used */
                              -1;
```

```
MK_ACHSENART                0,      /* Definition of the axis (bit coded) */
                            0,      /* Bit 0: 0 linear axis, 1 rotation axis */
                            0,      /* Bit 1: 0 observe HW limit switch, 1 ignore HW limit
                                    switch */
                            0,      /* Bit 3/2: 00 normal axis, 01 spindle */
                                    /* 10 Measurement axis, 11 Spindle & measurement axis */
                            0,      /* Bit 5/4: Setting rotation axes (Bit0 = 1) */
                            0;      /* 00 Rotation axis with absolute positioning */
                                    /* 01 Modulo 360°axis, sign determines direction */
                                    /* 11 modulo 360°axis, shortest distance is taken */
                                    /* Bit 7: handwheel */

MK_SPINDELART               1,      /* Assignment spindle handler-spindles */
                            1,      /* 1 = Standard spindle handler */
                            1;


/*-------------------------------------------*/
/*      2.  Software configuration           */
/*-------------------------------------------*/
MK_KUNDE                    "";     /* activates different customer-specific extensions */

MK_VOREINSTELLUNG           "";     /* for initialization after reset or program end */

MK_PASSWORT                 0;      /* password (no.) for machine constants editor */
                                    /* for integrated operating control*/

MK_SPRACHE                  0;      /* language selection for integ. operating control; 0=German
                                    */

MK_FLOPPYDISK               0;      /* no. of interface to which floppy is connected */
                                    /*, 0 = no floppy */

MK_DRUCKER                  0;      /* no. of interface to which printer */
                                    /* is connected, 0 = no printer */

MK_DRUCKER_V24MODE          9600,   /* settings for the printer interface */
                            8,      /* baud rate, data bits, parity, stop bits */
                            0,
                            1;

MK_CANOPEN_BAUDRATE         0,      /* CAN1 bit rate for CAN-Open, or 0 for SLIO */
                            1000;   /* CAN2 bit rate for CAN-Open drive */

MK_FEHLERRESTART            0;      /* 1 = restart after error allowed */

MK_S0T0_VERSATZ_ERLAUBT     0;      /* 1 = shift of S0 or T0 allowed */

MK_S_VERSATZSPERRE          0;      /* no. of the first protected workpiece coordinate */
                                    /* system or 0, if inactive */

MK_EDIT_SOFTKEYS            "GXYZMNCFP=IJRKE:P=()ABDHLOQSTUVW{}%";

MK_TEACHSTELLEN             3;      /* number of decimal positions which must be */
                                    /* taken into account during teaching*/
```

```
MK_TEACHHEADER                ””;      /* additional parameter for taught line */

MK_NCPROG_OHNE_KOMMENTARE     0;       /* 1 = store DIN programs without comment in Flash-PROM */

MK_NCPROG_NICHT_INS_EEPROM    0;       /* 1 = do not store DIN programs in Flash-PROM */

MK_NULLPUNKTE_SPEICHERN       0;       /* 1 = store zero points automatically */
                                       /* (only if CMOS-RAM is available) */

MK_METRISCH                   1;       /* 1 = distance and speed input */
                                       /* in metric system [mm] / [mm/min] */
                                       /* 0 = distance and speed input */
                                       /* in inch system [inch] or [inch/min] */

MK_CONST_REL_MM               1.0;     /* input resolution in metric */
                                       /* system, relating to 1 mm  */

MK_CONST_REL_INCH             25.4;    /* input resolution in inch system, */
                                       /* relating to 1 mm  */

MK_SW_ENDS_MIT_RAMPE          0;       /* 0 = in the case of SW limit switch, decelerate without
                                       ramp */
                                       /* 1 = decelerate with ramp */

MK_OVERRIDEMAX                1500;    /* max. permissible override [0.1%] */

MK_EPSILONMM                  0.01;    /* tolerance for positions (e.g. circle) [mm] */

MK_EPSILONGRAD                0.01;    /* tolerance for angles [Grad] ([degrees]) */

MK_KONTURFEHLER               0.01;    /* maximum permissible path error [mm] */

MK_LAH_GRENZWINKEL            90;      /* transition angle at which automatic */
                                       /* braking until standstill is to start */

MK_RADIUS_B_BEWERTUNG         1;       /* acceleration evaluation for circles or */
                                       /* spline arcs for speed reduction */

MK_HANDRADFILTER              0;       /* filter time const. for soft handwheel function in [ms] */

MK_MASCH_POLAR_KART           0;       /* 1 = polar machine coordinate system */

MK_KARTESISCH_ACHSNR          0,       /* no. of cos axis in the Cartesian coordinate syst. */
                              1,       /* no. of sin axis in the Cartesian coordinate syst. */
                              -1;      /* no. of the tangential correction axis or -1 */
MK_POLAR_ACHSNR               0,       /* no. of the radius axis in the pol. coordinate sys. */
                              1;       /* no. of the angle axis in the pol. coordinate sys. */

MK_DELTAT                     4;       /* rough interpolation cycle in [ms] */

MK_FIT_PRO_GIT                2;       /* number of fine interpolation cycles*/
                                       /* per rough interpolation cycle */
```

```
/* !!! The following 5 MCs are only active after the control        !!! */
/* !!! is restarted after the transfer of the MCs !!! */

MK_KANALANZAHL               1;       /* number of NC channels */

MK_PFELDGROESSE              2048;    /* parameter field size */

MK_LAH_VORLAUFTIEFE          256,     /* size of prebuffer in blocks for channel 0 */
                             0;       /* and for channel 1 */

MK_LAH_RUECKLAUFGRENZE       4,       /* no. of blocks that can be attributed to */
                             0;       /* the profile for channel 0 and channel 1 */

MK_SPV_SPEICHERGROESSE       1000;    /* size of intern. program memory in kByte */

MK_SPV_SYMBOLANZAHL          0;       /* max. number of symbols for symbolic */
                                      /* program number management */
MK_HEADERANZAHL              100;     /* max. number of NC programs to be managed */
                                      /* at the same time*/
MK_SPS_SPEICHERGROESSE       128;     /* size of the PLC program memory in kByte */

MK_SPS_DATENGROESSE          32;      /* size of the PLC data memory in kByte */
/*------------------------------   -----------*/
/*  3.  Setting of axes            */
/*------------------------------ ----------*/
/* resolution of position
encoder */
/* ---------------------------------------- */
MK_IMPULSE                   65536,   /* number of pulses per [MK_WEG] */
                             65536,   /* (after the quadruplication!) */
                             65536,
                             65536,
                             65536,
                             65536;

MK_WEG                       10,      /* specification in [mm] or [Grad] ([degrees])        */
                             10,      /* [WEG] ([POSITION]) at which the axis computer (!) */
                             10,      /* sees number of [MK_IMPULSE]          */
                             10,
                             10,
                             10;
MK_SSIKONF                   0,       /* configuration of SSI encoders */
                             0,       /* bit 0-6: no. of significant bits (12-32) */
                             0,       /* bit 7 : 1=binary code, 0=gray code */
                             0,
                             0,
                             0;

MK_ACHSEINGAENGE             1234,    /* assign. of limit switches to axis inputs */
                             1234,    /* 1234   */
                             1234,    /* input number of the spare input */
                             1234,    /* input number of the reference cam */
```

```
                        1234,    /* input number of the neg. limit switch */
                        1234;    /* input number of the pos. limit switch */

MK_US                   0,       /* monitor. of the position encoder protection signal */
                        0,       /* 1 = monitoring active, 0 = inactive */
                        0,
                        0,
                        0,
                        0;

MK_UA0                  0,       /* cable breakage monitoring, position encoder zero pulse */
                        0,       /* 1 = monitoring active, 0 = inactive */
                        0,
                        0,
                        0,
                        0;

MK_UA12                 0,       /* cable breakage monitoring, position encoder signal Ua1/Ua2
                                    */
                        0,       /* 1 = monitoring active, 0 = inactive */
                        0,       /* 2 = ignore nonrecurring fault */
                        0,
                        0,
                        0;

MK_TEACHACHSE           1,       /* marking of axes that are to be */
                        1,       /* transferred via the teach-in box  */
                        1,       /* during teaching */
                        0,
                        0,
                        0;

MK_MASSSTAB             1,       /* multiplication factor for the position */
                        1,       /* programmed in a DIN block  */
                        1,
                        1,
                        1,
                        1;

MK_HANDRADZUORDNUNG     0,       /* assignment of axes to handwheels */
                        0,       /* 0 = handwheel 0, 1 = handwheel 1 */
                        0,
                        0,
                        0,
                        0;

MK_HANDRADFAKTOR        1,       /* individual evaluation factor for the */
                        1,       /* handwheel funct., e.g. for special treatment */
                        1,       /* of rotation axes */
                        1,
                        1,
                        1;
```

```
MK_GRUNDOFFSET              0,      /* distance of the machine zero point*/
                            0,      /* from home position [mm] or [Grad] ([degrees]) */
                            0,
                            0,
                            0,
                            0;

MK_SYNCHRONOFFSET           0,      /* distance of the machine zero point between */
                            0,      /* synchronous axes after home position approach */
                            0,
                            0,
                            0,
                            0;

MK_SPINDELUMKEHRSPIEL       0,      /* spindle reverse backlash compensation */
                            0,      /* in [mm] or [Grad] ([degrees]) */
                            0,
                            0,
                            0,
                            0;

MK_SCHLEPPABSTAND           32000,  /* maximum following error in */
                            32000,  /* actual value encoder increments*/
                            32000,
                            32000,
                            32000,
                            32000;

MK_SCHLEPPZAEHLER           1,      /* number of fine interpolation cycles in which the */
                            1,      /* max. following error may be exceeded */
                            1,      /* until a following error is reported */
                            1,
                            1,
                            1;

MK_SCHLEPPGENAUHALT         0.001,  /* max. following error for exact positioning */
                            0.001,  /* with following error monitoring */
                            0.001,  /* in [mm] or [Grad] ([degrees]) */
                            0.001,
                            0.001,
                            0.001;

MK_GENAUHALTZEIT            0,      /* time [s] at which the following error must be smaller */
                            0,      /* than MK_SCHLEPPGENAUHALT */
                            0,
                            0,
                            0,
                            0;

MK_SYNCHRONABWEICHUNG       10,     /* max. permissible synchronous deviation from */
                            10,     /* synchronous axes in [mm] or [Grad] ([degrees]) */
                            10,
```

```
                        10,
                        10,
                        10;

MK_SW_ENDS_MINUS        -10000,  /* software limit stop (+)-direction */
                        -10000,  /* [mm], relating to basic offset  */
                        -10000,
                        -10000,
                        -10000,
                        -10000;

MK_SW_ENDS_PLUS         10000,   /* software limit stop (-)-direction */
                        10000,   /* [mm], relating to basic offset  */
                        10000,
                        10000,
                        10000,
                        10000;

/* ----------------- */
/* controller setting */
/* ----------------- */

MK_DRIFTABGLEICH        0,       /* offset adjustment [mV] for analog */
                        0,       /* axis interfaces */
                        0,
                        0,
                        0,
                        0;

MK_REGLER_MODE          2,       /* 0 = control loop open */
                        2,       /* 1 = reserved */
                        2,       /* 2 = P controller */
                        2,       /* 3 = P controller with feed-forward   */
                        2,       /* 4 = spindle  */
                        2;       /* 5 = PID controller with feed-forward */

MK_KP                   1,       /* KP of the P controller */
                        1,       /* (also for synchronous control) */
                        1,
                        1,
                        1,
                        1;

MK_TV                   0,       /* rate time [s] for D component of */
                        0,       /* PID controller (also for synchronous control) */
                        0,
                        0,
                        0,
                        0;

MK_TN                   0,       /* integral-action time [s] for integral action component of
                                    */
                        0,       /* PID controller (also for synchronous control) */
```

```
                        0,
                        0,
                        0,
                        0;

MK_KF                   0,      /* feed-forward factor */
                        0,      /* calculation: (Umax = setpoint for Vmax) */
                        0,      /*                Umax[0.1V] * 327.68                  */
                        0,      /* MK_KF = ------------------------------------------- */
                        0,      /*         Vmax [m/min]   MK_DELTAT     MK_IMPULSE[]   */
                        0;      / *        ------------ * ---------- * -------------   */
                                /*            60[]      MK_FIT_PRO_GIT    MK_WEG[mm]   */

MK_KB                   0,      /* factor for acceleration pilot control */
                        0,
                        0,
                        0,
                        0,
                        0;

MK_T2                   0,      /* filter time constant for fine polation [s] */
                        0,
                        0,
                        0,
                        0,
                        0;



/* ------------------------------------ */
/* settings for home position approach */
/* ------------------------------------ */
MK_REF_TYP              0,      /* type of home position approach */
                        0,      /* 0: with cam and zero pulse, */
                        0,      /* 1: with cam but without zero pulse, */
                        0,      /* 2: only reset counter with G74, */
                        0,      /* 3: only zero pulse, cam determines direction */
                        0;      /* 4: 2 cams, spare input on ARI is additional*/
                                /* reference cam and is approached first */

MK_REF_RICHTUNG_UND_FOLGE   1,  /* direction of axis at start of home pos. approach */
                        1,      /* and sequence for auto. home pos. approach */
                        1,      /*  0 => axis is not included */
                        1,      /*  X => pos. direction */
                        1,      /* -X => neg. direction */
                        1;

MK_REF_VMAX1            5,      /* speed to the cam [m/min] or [rpm] */
                        5,
                        5,
                        5,
                        5,
                        5;
```

```
MK_REF_BMAX1                1,      /* ramp for moving to the cam [m/sec²] or [rps²] */
                            1,
                            1,
                            1,
                            1,
                            1;

MK_REF_VMAX2                1,      /* speed from the zero pulse [m/min] or [rpm] */
                            1,
                            1,
                            1,
                            1,
                            1;

MK_REF_BMAX2                1,      /* ramp for moving from cam [m/sec²] or [rps²] */
                            1,
                            1,
                            1,
                            1,
                            1;



/* ----------------------- */
/* axis-related limit values */
/* ----------------------- */

MK_MODVMAX                  10,     /* modal axis speed */
                            10,     /* (manual traversing) [m/min] or [rpm] */
                            10,
                            10,
                            10,
                            10;

MK_VMAX                     20,     /* max. axis speed [m/min] / [rpm] */
                            20,
                            20,
                            20,
                            20,
                            20;

MK_BESCHL                   2,      /* acceleration ramp [m/sec²] / [rps²] */
                            2,
                            2,
                            2,
                            2,
                            2;

MK_BREMS                    2,      /* deceleration ramp [m/sec^2] or [rps^2] */
                            2,
```

```
                          2,
                          2,
                          2,
                          2;

MK_T_BESCHL               0,       /* damping time constant for acceleration ramps */
                          0,       /* and deceleration ramps [ms] */
                          0,
                          0,
                          0,
                          0;

MK_SPINDELMAX             6000,    /* spindle speed in [rpm] at 10 V */
                          6000,
                          6000;









/* ----------------------- */
/* path-related limit values */
/* ----------------------- */
MK_VBAHNMAX               20;      /* 20 max. material speed [m/min] */

MK_BAHNBESCHL             2;       /* acceleration ramp [m/sec2] */

MK_BAHNBREMS              2;       /* deceleration ramp [m/sec2] */

MK_T_BAHNBESCHL           0;       /*damping time const. deceleration/acceleration ramps [ms]*/

/*-----------------------------------------*/
/*  4.  Setting additional tabs            */
/*-----------------------------------------*/
MK_MESS_AUFLOESUNG        1.0,     /* resolution of the analog measured values [mm/V] */
                          1.0,
                          1.0,
                          1.0,
                          1.0,
                          1.0,
                          1.0,
                          1.0,
                          1.0,
                          1.0,
                          1.0,
                          1.0,
                          1.0,
                          1.0,
                          1.0,
                          1.0;
```

```
/*-------------------------------------------------*/
/*  5.  Technology - specific settings   */
/*-------------------------------------------------*/
MK_MFKT_UPR_TABELLE          0,       /* table of M functions after which a */
                             0,       /* G22 L9000+Mfktnr is to be added.  */
                             0,       /* the table may have a max. of 16 entries  */
                             0,
                             0,
                             0,
                             0,
                             0,
                             0,
                             0,
                             0,
                             0,
                             0,
                             0,
                             0,
                             0;
MK_WLK_C_GRENZWINKEL         45;      /* cutting/sewing: angle up to which the */
                                      /* C axis is to be abruptly fed */

MK_X_WINKEL                  0;       /* angle of the inclined X axis in */
                                      /* degrees (X-Z plane) */

MK_GEWINDE_VMAX              0;       /* thread grinding */

MK_SPINDELMAX_GS1            6000,    /* spindle speed in [rpm] at 10 V */
                             6000,    /* for gearbox stage 1 for spindle type 9 */
                             6000;

MK_SPINDELMAX_GS2            6000,    /* spindle speed in [rpm] at 10 V */
                             6000,    /* for gearbox stage 2 for spindle type 9 */
                             6000;

MK_SPINDELMAX_GS3            6000,    /* spindle speed in [rpm] at 10 V */
                             6000,    /* for gearbox stage 3 for spindle type 9 */
                             6000;

MK_SPINDELDREHZAHLMAX        6000,    /* max. permissible spindle speed in [rpm] */
                             6000,    /* (not taken into account by all */
                             6000;    /* spindle handlers.) */

MK_SPINDELDREHZAHLMIN        0,       /* min. permissible spindle speed in [rpm] */
                             0,       /* (not taken into account by all spindle handlers) */
                             0;

MK_TECHNOLOGIEDATEN1         0,       /* application-spec. technology param., which */
                             0,       /* lie in the parameter field from P760 */
                             0,
                             0,
                             0,
                             0,
```

```
                            0,
                            0,
                            0,
                            0;

MK_TECHNOLOGIEDATEN2        0,      /* like TECHNOLOGIEDATEN1 */
                            0,
                            0,
                            0,
                            0,
                            0,
                            0,
                            0,
                            0,
                            0;
MK_TECHNOLOGIEDATEN3        0,      /* like TECHNOLOGIEDATEN1 */
                            0,
                            0,
                            0,
                            0,
                            0,
                            0,
                            0,
                            0,
                            0;

MK_TECHNOLOGIEDATEN4        0,      /* like TECHNOLOGIEDATEN1 */
                            0,
                            0,
                            0,
                            0,
                            0,
                            0,
                            0,
                            0,
                            0;

/*-------------------------------------------*/
/*  6.  PLC - specific settings      */
/*-------------------------------------------*/

/* The following values are written in the data block 2 of the PLC */
MK_DW224_255                0,      /* DW 224 */
                            0,      /* DW 225 */
                            0,      /* DW 226 */
                            0,      /* DW 227 */
                            0,      /* DW 228 */
                            0,      /* DW 229 */
                            0,      /* DW 230 */
                            0,      /* DW 231 */
                            0,      /* DW 232 */
                            0,      /* DW 233 */
```

```
0,      /* DW 234 */
0,      /* DW 235 */
0,      /* DW 236 */
0,      /* DW 237 */
0,      /* DW 238 */
0,      /* DW 239 */
0,      /* DW 240 */
0,      /* DW 241 */
0,      /* DW 242 */
0,      /* DW 243 */
0,      /* DW 244 */
0,      /* DW 245 */
0,      /* DW 246 */
0,      /* DW 247 */
0,      /* DW 248 */
0,      /* DW 249 */
0,      /* DW 250 */
0,      /* DW 251 */
0,      /* DW 252 */
0,      /* DW 253 */
0,      /* DW 254 */
0;      /* DW 255 */
```

5 Interface PLC <-> NC operating system

The integrated PLC is a component of the hardware and software of the ETC control. It is programmed like a conventional PLC and generally has the same features. For the communication with the actual CNC control (NC), a formal interface is provided within the CNC, whose function follows the regulations according to IEC 550, ISO 4336 and VDI 3422.

In this chapter, the function of the interface is described. The individual signals are defined and their function mode described.

It is not necessary to describe the signal types and the timing of the signals, because the interface is created solely by internal memories and is thus subject to the same regulations as the data exchange within the CNC. Signal propagation delays must not be taken into account.

5.1 Definitions

The data exchange between NC computer and integrated PLC takes place via an internal RAM area, which is defined as ”Data words” for the PLC. The transfer takes place in data blocks ”DB0”, ”DB1”, ”DB2” and ”DB8 ... DB15”.

DB0: Contains the interface to the ETC_MMI.

DB1: Contains all standard signals (NC <-> PLC) as well as an area from the ”virtual keyboard”.

DB2: Interface between PLC and MMI.

DB8 ... DB15: This is where the interface data between the NC computer and the MMI is stored. This interface corresponds to the Dual-Port RAM interface, which is used as a connection to the MMI.

Every data block consists of 256 data words (DW) of 16 bit each. One block contains static signals as well as messages (data) that are exchanged with acknowledge characters (”handshake”). The data areas are bidirectional, i.e. the data from the PLC to the NC as well as the data from the NC to the PLC is included.

Note!

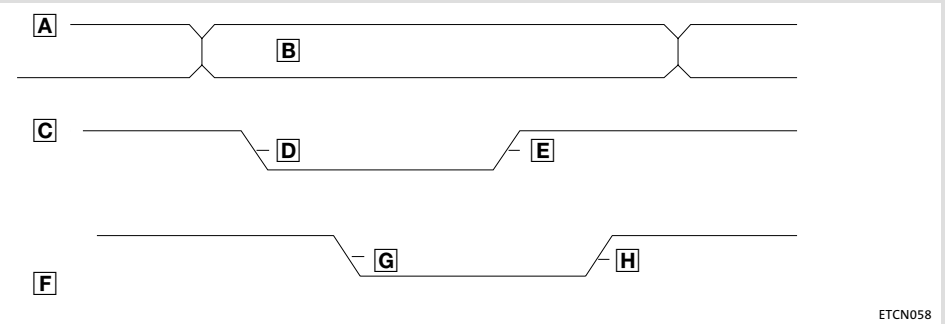
In CoDeSys, the signals/data of the data blocks are available as what is called system variables.

The data exchange between NC and SPS takes place cyclically during each cycle of the PLC program. The data which is transferred from the NC to the PLC, is copied *before* a PLC cycle starts, the data from the PLC to the NC, *after* a cycle.

Static signals are set without prior evaluation. Data whose evaluation must be ensured is transmitted as a message and has two additional signals for a handshake: ”Strobe” and ”Acknowledgement”.

With ”Strobe”, the transmitter validates the previously written data, with ”Acknowledgement”, the receiver indicates the evaluation of the data. If the transmitter detects the acknowledgement of the receiver, it withdraws the ”Strobe” signal, after which the receiver deletes its ”Acknowledgement” signal. It is now permitted to transfer new data. For the data transfer, no special temporal conditions must be observed, but make sure that the signals occur in order described. The times of the ”Strobe” and ”Acknowledgement” signals are monitored. They must not exceed a period of one second. If this period is exceeded, an error message is output, which leads to the termination of the currently running DIN program.

In general, the data exchange for messages always follows the following signal pattern:



A **Data**
B New value
C **Strobe**
D Valid
E Acceptance detected
F **Acknowledgement**
G New value accepted
H Next value detected

In the following description of the signals and data, the respective signal type is defined as

- static
- message
- strobe
- acknowledgement

The designation of the data words is carried out according to the following procedure:

Dxxx.yy

xxx = number of the data word in a data block (0 ... 255)

yy = bit number in a data word (0 ... 15)

5.1.1 Data block 0

When using DB0, bear in mind that the data may only be evaluated in the 2nd PLC task (PLC_PRG2/ OB20). This is necessary because the data is updated in DB0 before the 2nd task is called and the NC evaluates the data after execution. The error interface is an exception as it is synchronized by a strobe (error counter) and an acknowledgement signal (error acknowledgement).

Assignment

Data word	CoDeSys system variable	Direction
001.00 – 001.15	db0_sps2nc_nc_fehlerquittung_w	PLC → NC
016.00 – 095.15	db0_nc2sps_fehlertext_s	PLC ← NC
096.00 – 127.15	db0_nc2sps_fehlerinfo_s	PLC ← NC
128.00 – 128.15	db0_nc2sps_aktuelles_menu_w	PLC ← NC
129.00 – 129.15	db0_nc2sps_ob20_ereignis_w	PLC ← NC
132.00 – 132.15	db0_nc2sps_funktionsstatus_w	PLC ← NC
133.00 – 133.15	db0_nc2sps_nc_fehlerzaehler_w	PLC ← NC
134.00 – 134.15	db0_nc2sps_nc_fehlerklasse_w	PLC ← NC
135.00 – 135.15	db0_nc2sps_nc_fehlermodul_w	PLC ← NC
136.00 – 136.15	db0_nc2sps_nc_fehlernummer_w	PLC ← NC
137.00 – 137.15	db0_nc2sps_aktuelles_untermenu_w	PLC ← NC
148.00 – 255.15	db0_nc2sps_g253_textausgabe_s	PLC ← NC

All of the areas that are not listed explicitly are reserved.

Description of the signals

Data word	Name	Direction	Type of signal
001	Error acknowledgement	PLC → NC	static

This data word can be used by the PLC to recognize error messages and to acknowledge error messages if the PLC has to evaluate an output error message. If DW001 is unequal to DW133 (error counter), a new error message is pending. Once the PLC has evaluated the error message, DW001 must be set to the value of DW133. This way, the error in the NC computer is acknowledged.

Data word	Name	Direction	Type of signal
016.00 – 095.15	Error text	NC → PLC	static

From DW016 onwards, the error text of the currently pending error is stored as null-terminated string.

Data word	Name	Direction	Type of signal
096.00 – 127.15	Additional error information	NC → PLC	static

From DW096 onwards, additional information (e.g. axis letter) on the currently pending error is stored, if applicable, as a null-terminated string.

Data word	Name	Direction	Type of signal
128	Current menu	NC → PLC	static

This is where the current menu is identified:

0 = invalid
1 = main menu
2 = set-up
3 = automatic
4 = programming
5 = diagnostics
6 = user

If you want the PLC program to change to a standard menu, the identification of the new menu must be stored in DW128 prior to executing the function ”New state()”.

Data word	Name	Direction	Type of signal
129	Event	NC → PLC	static

This is where the NC stores an identification, for which reason the 2nd PLC task is to be carried out, before the 2nd PLC task is called (PLC_PRG2/ OB20). This is to ensure a corresponding reaction can be triggered. The contents are updated in each PLC cycle and do not have to be reset.

0 = cyclic call
1 = key message => evaluate DW130
2 = input termination => evaluate DW131
3 = new machine constants were loaded
4 = key message during active input => evaluate DW130
5 = block search limit reached
6 = block search target position reached

Data word	Name	Direction	Type of signal
132	State after termination of function	NC → PLC	static

Each called function (Ba_Einrichten(), Ba_Automatik(), etc.) reports back a state here:

0 = operating mode still active.
1 = operating mode no longer active. System must be set to the initial state (main menu).

This state must be checked by the PLC program after the function has been terminated. If DW132 has the value 1, it means that the initial state was set in the function that was terminated previously.

Data word	Name	Direction	Type of signal
133	Error counter	NC → PLC	static

DW133 is used by the NC computer in order to report errors to the PLC. For this purpose, DW133 is incremented and is thus unequal to DW001 (error acknowledgement). After the error message has been evaluated, DW001 must also be incremented by the PLC. This way, the NC computer recognizes the error acknowledgement and can transmit another error message, if required.

Data word	Name	Direction	Type of signal
134	Error class	NC → PLC	static

This is where the error class of each reported error is specified. The following error classes exist:

1 = slight local error
2 = fatal local error
3 = slight global error
4 = fatal global error

Local errors only affect the sub system, which has reported the error. Global errors always stop or interrupt the currently running DIN program.

Data word	Name	Direction	Type of signal
135	Error module	NC → PLC	static

This is where the module which has reported the error is specified for every reported error. The following modules can report errors:

1 = axis computer
2 = initialization
3 = interpreter
4 = coupling
5 = PLC runtime system
6 = rough interpolator
7 = memory management
9 = central control
10 = PLC
11 = operating system
12 = CAN
13 = auxiliary routines
14 = communication NC -> MMI
15 = communication MMI -> NC
18 = DS402 conform drive amplifier
19 = Lenze drive amplifier

Data word	Name	Direction	Type of signal
136	Error number	NC → PLC	static

This is where the actual error number is reported. An error number is always assigned to the corresponding error module. The error can only be clearly determined by means of the error number **and** error module.

Data word	Name	Direction	Type of signal
137	Current submenu	NC → PLC	static

reserved

Data word	Name	Direction	Type of signal
148.00-255.15	Text output G253	NC → PLC	static

The texts that are output with the G function G253 (without parameter E) are shown from DW148 onwards for the PLC.

5.1.2 Data block 1

In data block 1 the standard signals are exchanged. Standard signals are all signals and messages recorded in the respective specifications, which reflect the state of the PLC and the NC and affect the operation of the NC or the PLC. They are also signals/messages that do not exist in the general specifications but are part of the standard scope of the ETC.

In the following, you can find an overview of the assignment of the data block. This is followed by individual descriptions of signals and messages.

Assignment

Data word	Name	Direction
000.00	db1_sps2nc_notaus_bit	PLC → NC
000.01	db1_sps2nc_vorschubfreigabe_bit	PLC → NC
000.02	db1_sps2nc_soforthalt_bit	PLC → NC
000.03	db1_sps2nc_einzelfunktionsperre_bit	PLC → NC
001.00 – 001.15	db1_sps2nc_vorschubfreigabe_w	PLC → NC
002.00 – 002.15	db1_sps2nc_position_halt_w	PLC → NC
003.00 – 003.15	db1_sps2nc_verfahrtastenfreigabe_plus_w	SP → NC
004.00 – 004.15	db1_sps2nc_verfahrtastenfreigabe_minus_w	PLC → NC
005.00 – 005.15	db1_sps2nc_reglerfreigabe_w	PLC → NC
007.00	db1_sps2nc_einlesefreigabe_bit	PLC → NC
009.00 – 009.07	db1_sps2nc_programmstart_b	PLC → NC
009.08 – 009.11	db1_sps2nc_use_spsoverride_b	PLC → NC
012.00	db1_sps2nc_programmstop_b	PLC → NC
012.08	db1_sps2nc_unterbrechen_bit	PLC → NC
013.00 – 013.07	db1_sps2nc_einzel_folgesatz_bit	PLC → NC
013.08 – 013.15	db1_sps2nc_satzausblenden_bit	PLC → NC
014.00	db1_sps2nc_rueckzug_bit	PLC → NC
016.00 – 017.15	db1_sps2nc_tastensignale_aw	PLC → NC
018.00 – 021.15	db1_sps2nc_qbit_signale_aw	PLC → NC
031.00 – 031.15	DB1_sps2nc_freigaben_mmi_w	PLC → NC
032.00	db1_sps2nc_mfkt_quitt_bit	PLC → NC

Data word	Name	Direction
033.00	db1_sps2nc_hfkt_quitt_bit	PLC → NC
035.00	db1_sps2nc_extsync_enable_bit	PLC → NC
035.08	db1_sps2nc_programmhalt_aktiv_bit	PLC → NC
036.00 – 037.15 038.00 – 039.15	db1_sps2nc_qin_mask_aw db1_sps2nc_qout_mask_aw	PLC → NC PLC → NC
042.00 – 043.15 044.00 – 045.15	db1_sps2nc_qin_offset_ab db1_sps2nc_qout_offset_ab	PLC → NC PLC → NC
080.00 – 080.15 081.00 – 081.15 082.00 – 082.15 083.00 – 083.15	db1_sps2nc_spsoverride_allg_w db1_sps2nc_spsoverride_spindel_w db1_sps2nc_spsoverride_ozillation_w db1_sps2nc_spsoverride_sps_w	PLC → NC PLC → NC PLC → NC PLC → NC
084.00 – 091.15	db1_sps2nc_verfahr_ab	PLC → NC
128.00 128.01	db1_nc2sps_betriebsbereit_1_bit db1_nc2sps_betriebsbereit_2_bit	PLC ← NC PLC ← NC
129.00 – 129.15	db1_nc2sps_referenzpunkt_w	PLC ← NC
130.00 – 131.15	db1_nc2sps_referenzfahrt_laeuft_w	PLC ← NC
131.00 – 131.15	db1_nc2sps_betriebsart_w	PLC ← NC
132.00 132.02 132.03 132.04	db1_nc2sps_nc_programm_laeuft_bit db1_nc2sps_satzvorlauf_aktiv_bit db1_nc2sps_einzelsatz_laeuft_bit db1_nc2sps_unterbrechen_aktiv_bit	PLC ← NC PLC ← NC PLC ← NC PLC ← NC
132.08	db1_nc2sps_alle_achsen_stehen_bit	PLC ← NC
133.00 – 133.07	db1_nc2sps_wahlweise_halt_b	PLC ← NC
133.08 – 133.15	db1_nc2sps_programmstartzaehler_b	PLC ← NC
134.00 – 134.15	db1_nc2sps_achse_faehrt_w	PLC ← NC
135.00 – 135.15	db1_nc2sps_modal_am_ziel_w	PLC ← NC
137.00 – 137.07	db1_nc2sps_programmstart_b	PLC ← NC
137.08 – 137.15	db1_nc2sps_einzel_folgesatz_bit	PLC ← NC
138.00 – 138.15 139.00 – 139.15	db1_nc2sps_canmodul_vorhanden_w db1_nc2sps_can_status_w	PLC ← NC PLC ← NC
142.00	db1_nc2sps_programmstop_b	PLC ← NC
144.00	db1_nc2sps_vorschub_halt_b	PLC ← NC
150.00 151.00 – 151.15	db1_nc2sps_mfkt_strobe_bit db1_nc2sps_mfkt_w	PLC ← NC PLC ← NC
159.00 160.00 – 160.15	db1_nc2sps_hfkt_strobe_bit db1_nc2sps_hfkt_w	PLC ← NC PLC ← NC
177.00 – 177.15	db1_nc2sps_refpunkt_angefahren_w	PLC ← NC
178.00 – 178.15 179.00 – 179.15 180.00 – 180.15 181.00 – 181.15	db1_nc2sps_endschalter_plus_w db1_nc2sps_endschalter_minus_w db1_nc2sps_referenznocken_w db1_nc2sps_reserveeingang_w	PLC ← NC PLC ← NC PLC ← NC PLC ← NC
182.00 – 182.15 183.00 – 183.15 184.00 – 184.15 185.00 – 185.15	db1_nc2sps_slave_endschalter_plus_w db1_nc2sps_slave_endschalter_minus_w db1_nc2sps_slave_referenznocken_w db1_nc2sps_slave_reserveeingang_w	PLC ← NC PLC ← NC PLC ← NC PLC ← NC
192.00 – 199.15	db1_nc2sps_copmodul_vorhanden_aw	PLC ← NC
200.00 – 200.15 201.00 – 201.15 202.00 – 202.15 203.00 – 203.15	db1_nc2sps_spsoverride_allg_w db1_nc2sps_spsoverride_spindel_w db1_nc2sps_spsoverride_ozillation_w db1_nc2sps_spsoverride_sps_w	PLC ← NC PLC ← NC PLC ← NC PLC ← NC
204.00 – 211.15	db1_nc2sps_verfahr_ab	PLC ← HMI
212.00 – 219.15	db1_nc2sps_mmi_tasten_aw	PLC ← HMI

All of the areas that are not listed explicitly are reserved.

Description of the signals

Data word	Name	Direction	Type of signal
000.00	EMERGENCY STOP	PLC → NC	static

0 = emergency stop state
1 = normal operating state

Effect in the NC: The 0 signal interrupts all movements, the program process is interrupted.

Data word	Name	Direction	Type of signal
000.01	Feed enable (total)	PLC → NC	static
001.00-15	Feed enable axis 0 ... 15	PLC → NC	static

For each axis a ”Feed enable” signal is output. There is also a common enable signal. The resulting feed enable is created as a logic AND operation with the axis-dependent feed enables.

Signal state ”1” activates feed enable.

Effect in the NC: The ”1” signal enables the respective axis to move if the ”Feed stop” signal is not set.

A ”0” signal stops the respective axis or all of the axes. Moving axes are stopped with the set deceleration ramp.

In the case of interpolation, all of the participating axes are prevented from approaching, or they are decelerated if feed enable is rejected for one of the participating axes.

In manual operation, the traversing commands are interrupted, i.e. after the feed enable is restored, the command must be given again in order to continue.

In automatic operation, the movements are only interrupted, the traversing command continues as soon as the feed enable is restored.

Data word	Name	Direction	Type of signal
000.02	Quick stop	PLC → NC	static

A common signal for all axes:

0 = The movement of axes is enabled.
1 = Moving axes are stopped *without* deceleration ramp (”quick stop”).

Effect in the NC: The ”1” signal causes the moving axes to stop without deceleration ramp. The movements are only interrupted and the traversing command continues as soon as the signal is reset to 0.

Application: Response to ”Emergency stop” in applications, which require the program to continue after an emergency stop. In this case, only the feed enable is canceled by the PLC. If the axes should still stop abruptly, you can set the ”Quick stop” signal beforehand.

Data word	Name	Direction	Type of signal
000.03	MMI individual function inhibit	PLC → NC	static

With this signal, all individual functions (e.g. G functions, M functions, ...) of HMI can be inhibited.

Effect in the NC: The ”1” signal has the effect that no individual block of the MMI is executed via the NC.

Application: The signal could be used for inhibiting traversing functions which are triggered via an individual block on the HMI, as long as the home positions of the axes are not known.

Data word	Name	Direction	Type of signal
002.00-15	Position stop axis 0 – 15	PLC → NC	static

A signal can be output for each axis in order to stop it. In the case of signal state ”1” the axis stops at its position, in the case of signal state ”0” positioning is allowed.

The signal is set to ”0” by default.

Effect in the NC: Signal state ”1” has the effect that the respective axis is not positioned, irrespective of the feed enable. Within the control, the axis is treated as though it is traveling. Only the actual positioning is suppressed. The actual values are generated internally.

In the case of signal state ”0”, the axis is moved without restrictions, depending on other conditions. The signal is axis-specific, the other axis are not affected. This also applies to path operation.

The signal should only change when the axis is stationary. If the axes are moving, the movement is stopped immediately without a ramp.

Application: Testing mode during which specific axes must not move.

Data word	Name	Direction	Type of signal
003	Enable traverse keys (+) Axis 0 - 15	PLC → NC	static
004	Enable traverse keys (-) Axis 0 - 15	PLC → NC	static

For each axis, an enable signal for the manual traverse keys is output. Signal state ”1” activates the enable signal. The signal is only output in ”Manual operation” or ”Interrupt” if all conditions for the manual traversing of the axes are met. The signal is does not depend on feed or controller enable.

Effect in the NC: The command of the manual traverse key (see description DB15) is only executed if the enable signal of the PLC is output. A positive edge of the signal while a command is executed does not have any impact. A negative edge cancels the execution.

Application: The signal is important if the traverse keys are not realized via the PLC but via the ”virtual keyboard” (computer).

The enable values control the effect of the traverse keys. In the case of value 0, the traverse keys from data block 1 and data word 84 ... 91 are considered by the PLC. In the case of value 1, the traverse keys which are realized via the ”virtual keyboard” (MMI) are considered.

Data word	Name	Direction	Type of signal
005.00-15	Controller enable axis 0 - 15	PLC → NC	static

For each axis, a ”Reglerfreigabe” (”Controller enable”) signal is output. Signal state ”1” activates the controller enable. The controller enable is output before the feed enable is activated.

Effect in the NC: In ”0” state, the position control loop of the respective axis is opened. Any current movements and program processing is interrupted; the axes are stopped immediately (without ramp).

In ”1” state, the control loops are closed. When the control loops close, the current actual positions of the axes are transferred to the NC computer.

Data word	Name	Direction	Type of signal
007.00	Read enable	PLC → NC	static

The ”1” signal enables the execution of the next block.

Effect in the NC: If the signal has state ”1”, the next NC block can be executed.

State ”0” inhibits the processing of the next traversing block in the control (other blocks, such as arithmetic blocks, are processed!) as well as the output of synchronized M functions to the PLC.

If the state switches to ”0”, the current block is not canceled.

In manual operation, the processing of a new data record as well as the transfer of M functions is inhibited.

Data word	Name	Direction	Type of signal
009.00-07	Program start	PLC → NC	static

The starting signal is a byte information which can adopt a value between 0 and 255.

The program start requirement is output if the start signal of the HMI to the PLC (DW137) has a value unequal to ”0” or a corresponding input of the PLC has been set (”Press the start key”) and the necessary starting conditions have been met.

The value is set to ”0” when the ”NC program is running” signal is detected.

The information is divided in two half bytes (nibble). The nibble with the higher value holds information on the type of program to be started; the one with the lower value the mode, in which the start is to be executed. (for list of meanings, see DW137).

Note!

The information is in general taken from the start signal of the HMI. In special cases of application (direct start via PLC), the PLC itself generates the information.

Effect in the NC: Value ”1” of the signal triggers the defined program in the corresponding mode.

If entries are to be made in a P field, they must be entered beforehand (see DW137). The ”NC program is running” signal of the corresponding channel is set to ”1”.

Data word	Name	Direction	Type of signal
009.08-11	Enable override	PLC → NC	static

Controls the effect of the override keys (PLC or HMI).

Effect in the NC: In the case of value ”1”, the overrides set by the PLC in data block 1 and data word 80-83, or the MMI overrides set via the ”virtual keyboard”, are considered.

Data word	Name	Direction	Type of signal
012.00	Program stop	PLC → NC	static

The ”Stop” signal is set to value ”1” if the stop signal from HMI to the PLC (DW142) has value ”1” or if a corresponding input of the PLC is set (”Press the stop key”) or if the conditions for further processing of a program are not met due to other reasons.

The value is set to ”0” if the ”NC program is running” or ”NC program is running 1” signal adopts the value ”0” in ”Interrupt” state.

Effect in the NC: The value ”1” has an effect in the NC that depends on the set operating mode.

Manual operation: The current movement is interrupted; any other pending tasks are canceled. The ”NC program is running” signal is set to ”0”.

Automatic: Termination of the currently running NC program. The NC adopts the initial state. The ”NC program is running” signal is set to ”0”.

Data word	Name	Direction	Type of signal
012.08	Interrupt	PLC → NC	static

With the pos. edge of the interrupt signal, a currently running program or a program which has been stopped by a 3 class error can be interrupted (Note: A program is stopped in the case of a class 3 error if the machine constant MK_FEHLERRESTART is set to ”1”. Otherwise, the program is canceled.). In this state, the axes can be moved manually and individual functions or other programs can be executed.

Once programs and individual functions are inactive, the interrupt state can be exited by means of a pos. edge of the interrupt signal. The interrupted program can then be resumed by triggering a normal start or canceled by pressing the stop key. When the program is resumed, a balancing run of the axes will take place at first taking the shortest route to the point of interruption with 1/10 of the max. traversing speed.

The interrupt signal is ORed with the interrupt signal in the virtual keyboard.

Effect in the NC: NC switches to interrupt state with the 1st pos. edge. With the next pos. edge the state is exited again if no other actions are active.

Note!

In the interrupt state it is not possible to make coordinate shifts or zero point corrections.

Data word	Name	Direction	Type of signal
013.00 - 07	Individual/following block	PLC → NC	static

Switching between following and individual block operation. (This signal is ORed with the ”Individual/following block” signal from the DB15.)

Effect in the NC: A program is processed continuously (following block) or block by block (individual block):

0 = following block
1 = individual block

Data word	Name	Direction	Type of signal
013.08 - 15	Hide block	PLC → NC	static

With this signal, the ”Hide block” function can be switched on or off. (This signal is ORed with the ”Hide block” signal from the DB15).

Effect in the NC: Blocks with a preceding ”/” are hidden, i.e. they are not executed:

0 = Hide block inactive
1 = Hide block active

Data word	Name	Direction	Type of signal
014.00 - 07	Return	PLC → NC	static

The signal (a bit per NC channel) makes it possible to return (”Return”) to the programmed path. During this process, the return movement is not limited by blocks without geometry, i.e. individual S/T/M/Q functions, coordinate shifts and G21 blocks are permitted and do not result in the limitation of the return memory.

All other G functions, which are not pure interpreter functions, (📖 91)result in the limitation of the return memory as usual. During the return, M functions are output as they have been programmed if no other customer-specific solution has been planned. G21 and Q functions are ignored during the return.

Effect in the NC: When the signal is set, the NC processes the blocks in the prebuffer in reverse order. During this process, traversing blocks are output with negative speed.

Data word	Name	Direction	Type of signal
018.00	Q bit signal 0	PLC → NC	static
...	...	...	...
021.15	Q bit signal 63	PLC → NC	static

Application and program-specific signals for controlling the program flow in the NC control. The meaning of the values must be defined individually. The values depend on the inputs of the PLC and internal links.

Effect in the NC: The signals are stored unchanged in the internal ”Q-Feld” (”Q field”) as Q0 ... Q63. Individual entries of the Q field can be accessed in the program to ensure that it is possible to control the program flow.

Fast inputs: If fast inputs are configured (see DW36.0 ... 37.15), the corresponding Q bits are not read from the Q field but directly from the corresponding inputs E0.0 ... E3.7. This makes it possible to react more quickly to an external event as it is not diverted via the integrated PLC.

Data word	Name	Direction	Type of signal
031.00 - 15	Enable MMI	PLC → NC	static

Up until now this data word is only used in connection with the standard HMI. It is used to signal MMI which PLC functions are supported by the loaded PLC (coded in bits).

Bit number	Meaning (PLC support)
0	Teaching
1	Block search
2	Interrupt
3	Version output
4	Key control

By setting bits 0 ... 2, a corresponding soft key is shown on the HMI. If bit 3 is set, HMI requests a version string via the message interface in the DB2. If bit 4 is set, PLC is in control of the keys via the PLC keys provided in the ETC-MMI (📖 268).

Effect in the NC: none

Data word	Name	Direction	Type of signal
032.00	Acknowledgement for M function	PLC → NC	Acknowledgement

The signal is set to ”0” when the PLC has detected the value ”0” of the strobe for M functions and accepted the data of the M functions transferred by the NC. It is set to ”1” when the strobe has accepted the value ”1” again (📖 240).

Effect in the NC: If the signal has the value ”1”, new data can be transferred to the PLC; in the case of value ”0”, the PLC has accepted the pending data. In this case, the strobe is set from ”0” to ”1” for signaling that the acceptance was successful. When the signal changes from ”1” to ”0”, the NC starts to process the next block provided that the ”Read enable” has been activated.

Data word	Name	Direction	Type of signal
033.00	Acknowledgement for H function	PLC → NC	Acknowledgement

The signal is set to ”0” when the PLC has detected value ”0” of the strobe for H functions and accepted the data of the H functions which has been transferred by the NC. It is set to ”1” when the strobe has accepted value ”1” again.

Effect in the NC: If the signal has the value ”1”, new data may be transmitted to the PLC, in the case of value ”0”, the PLC has accepted the pending data. In this case, the strobe is set from ”0” to ”1” by the NC (acceptance signal has been detected). If the data acceptance takes longer (>10 sec.), the ”Read enable” signal must be canceled prior to setting the acknowledgement signal.

When the signal switches from ”1” to ”0”, the NC starts to process the next block if ”Einlesefreigabe” ”Read enable” has been activated.

Data word	Name	Direction	Type of signal
035.00	Enable ext. synchronization signal	PLC → NC	Enable

Meaning: reserved

Effect in the NC: reserved

Data word	Name	Direction	Type of signal
035.08	Program stop active	PLC → NC	static

The PLC uses ”1” to signal to the HMI that the process of a NC program has been stopped.

Effect in the NC: none

Data word	Name	Direction	Type of signal
036.00-037.15	Enable of the ”fast inputs”	PLC → NC	static

The PLC uses a ”1” to specify the corresponding digital input as the ”fast input”, i.e. the input signal is evaluated directly by the rough interpolator (in rough interpolation cycle). If enable is reset again by the PLC, it can immediately use the inputs again exclusively.

Note!

The state of the ”fast input” depends on an OR operation of the Q bit signal of the PLC and the state of the digital input.

Use the data words ”Offset of fast inputs” to position the enable (mask) in any area of the input image as ”fast input”. This way, any digital input can be specified as ”fast input”.

Effect in the NC: By means of corresponding DIN programming (G150, G151) it is possible, to have time-critical digital inputs evaluated directly by the rough interpolator as Q bits, while avoiding the PLC.

Example: With the following lines, the inputs %IX64.1, %IX64.4, %IX64.13 and %IX65.7 are configured as fast inputs.

DB1_SPS2NC_QIN_OFFSET_AB[0] := 64;

DB1_SPS2NC_QIN_OFFSET_AB[1] := 65;

DB1_SPS2NC_QIN_MASK_AW[0] := 2#0010000000010010;

DB1_SPS2NC_QIN_MASK_AW[1] := 2#0000000010000000;

Data word	Name	Direction	Type of signal
038.00-039.15	Enable the ”fast outputs”	PLC → NC	static

Value ”1” withdraws the control of the PLC over the corresponding output and assigns it to the rough interpolator; i.e. an output enabled as ”fast output” can only be operated by the rough interpolator. As soon as the PLC withdraws the enable command, it can use the outputs again for itself; the output signal states are immediately reflected on the outputs.

Use the data words ”Offset of fast outputs” to position the enable (mask) in any area of the output image as ”fast output”. This way, any digital output can be specified as ”fast output”.

After the control has been taken over, the last states remain intact.

Effect in the NC: By means of corresponding DIN programming (parameter Q), it is possible have time-critical digital outputs operated directly by the rough interpolator (in rough interpolation cycle).

Data word	Name	Direction	Type of signal
042.00-043.15	Offset of the ”fast inputs”	PLC → NC	static

Use the two data words to position the 2 masks ”Enable fast inputs” to any place in the process image.

Effect in the NC: The corresponding inputs are made available to the NC as Q bits.

Example: see ”Enable of fast inputs”

Data word	Name	Direction	Type of signal
044.00-045.15	Offset of the ”fast outputs”	PLC → NC	static

Use the two data words to position the 2 masks ”Enable fast outputs” anywhere in the process image. By default, the enable of the ”fast outputs” affects the 1st digital I/O card (EC-IO).

Effect in the NC: The Q bits are transferred to the corresponding outputs.

Data word	Name	Direction	Type of signal
080	General override for axes	PLC → NC	static
081	Override for spindle speed	PLC → NC	static
082	Override for oscillation speed	PLC → NC	static
083	Override for PLC axes (Target position approach)	PLC → NC	static

Definition of a signed evaluation factor for the currently valid traversing speed. The specification is made in steps of 0.1 %.

Effect in the NC: The selected value is used as factor for evaluating the currently programmed maximum speed by the rough interpolator.

The override is effective in manual as well as automatic operation. It can be inhibited by M functions within a program (M48/M49 override on/off).

Data word	Name	Direction	Type of signal
084.00-07	Traverse key axis 0	PLC → NC	static
084.08-15	Traverse key axis 1	PLC → NC	static
...	...	...	...
...	...	...	....
091.00-07	Traverse key axis 14	PLC → NC	static
091.08-15	Traverse key axis 15	PLC → NC	static

These signals are used for traversing the axes. One byte is available for each axis. By writing the bytes, the respective action is executed. These traverse keys are only active if traverse keys enable is not set.

Effect in the NC:

-100 ... +100: traversing with % of the maximum speed. The sign defines the direction.

+/-101: inching by increments (corresponding to DB15 DW44 ... 47) on the path if there is no feed enable. Only available for a programmed traversing movement.

+/-102: home position approach (sign is ignored)

+/-103: setting zero point (sign is ignored)

+/-104: step travel (P673 increment)

+/-105: handwheel active - evaluation factor 1

+/-106: handwheel active - evaluation factor 3

+/-107: handwheel active - evaluation factor 10

+/-108: handwheel active - evaluation factor 30

+/-109: handwheel active - evaluation factor 100

+/-110: traverse command for PLC axis

+/-111: traversing with the speed from P208-223.

The sign specifies the direction.

Description of the traverse command for PLC axes: The PLC can move any axis to a specific position that has been entered in the P field. The following parameters are used for this purpose:

P176 ... 191 = target position specification by PLC for axes 0 ... 15 [mm] or [°]

P208 ... 223 = speed specification by PLC for axes 0 ... 15 [mm/min] or [rpm]

P240 ... 255 = speed specification by PLC for axes 0 ... 15 [m/sec$^{2]}$ or [rps^2]

The following parameters are available for display:

P144 ... 159 = accepted target position after positive edge of the traverse key [mm] or [°]

P160 ... 175 = current modal actual position [mm] or [°]

A target point approach of an axis starts when the value +/-110 is specified in its traverse key. Make sure in the process that the NC computer detects the change of the traverse key (if required, observe handshake; see diagram).

This is precisely when the target point is accepted. (Note: Also a change from 110 to –110 or vice versa has the effect that the target position is accepted again.) The axis moves for as long as the value +/-110 is specified in the traverse key and the target has not yet been reached. A change to the speed specification (P208 ... 223) is also accepted when the axis is in motion. When the target point is reached, the NC sends the ”PLC axis reached target” signal in the DB1.

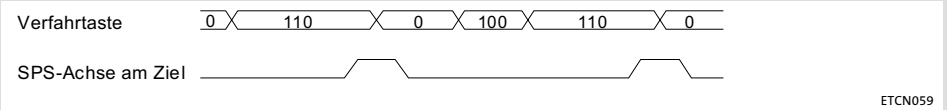
Travelling generally takes place via the ”modal offset”, i.e. parallel to any programmed NC target position. When the target position specification is accepted from the P field, a distinction is made between two scenarios depending on the configuration of the traverse key:

1. G122 X0 (default configuration)

The modal target position results from the difference between target position specification of the PLC and actual NC actual position. The specification therefore takes effect in relation to the current coordinate system.

2. G122 X1

The modal target position results directly from the target position specification of the PLC. The specification therefore takes effect in relation to the current NC actual position.



Data word	Name	Direction	Type of signal
128.00	NC-Betriebsbereit_1	NC → PLC	static

Value ”1” indicates that the NC control is ready for operation in all operating modes. The ”0” signal appears when the NC control changes to an error state which can be remedied and which interrupts the program flow (the ”NC program is running” signal remains on ”1”).

The signal changes from ”0” to ”1” after the cause of the errors has been remedied (error acknowledged by operator).

Effect in the PLC: not defined

Data word	Name	Direction	Type of signal
128.01	NC-Betriebsbereit_2	NC → PLC	static

Value ”1” indicates that the NC control is ready for operation in all operating modes. The ”0” signal appears when the control changes to an error state which cannot be remedied and which cancels the program flow or is in ”emergency stop” state. The signal changes from ”0” to ”1”, after the cause of the error has been remedied (error acknowledged by operator) and/or the control is not in the ”emergency stop” state.

Effect in the PLC: not defined

Example Change to emergency stop state, controller enable is withdrawn

Data word	Name	Direction	Type of signal
129.00-15	Home position axis 0 – 15	NC → PLC	static

The signal has the value ”0” after the control has been switched on and if the home position is unknown. The signal has the value ”1” if the home position of the respective axis has been approached or if value -1 has been entered for machine constant MK_REF_RICHTUNG_UND_FOLGE for the respective axis.

Effect in the PLC: Checking the home position approach and disabling machine functions or traversing movements (feed enable) if the home position is not approached.

Data word	Name	Direction	Type of signal
130.00-15	Homing is running	NC → PLC	static

The signal has the value ”1” if the homing of the respective axis is currently running.

Effect in the PLC: Checking whether the home position approach of an axis is currently active.

Data word	Name	Direction	Type of signal
131	Operating mode	NC → PLC	static

The operating mode word shows the PLC the operating mode selected by the operator, in which the NC control is at present:

0 = initial state; no axis movement possible
1 = set-up
2 = set-up functions: modal traveling, step travel, zero points
3 = automatic, autom. processing of programs (individual block, following block)
4 = programming
80H = diagnostics (DW131.7)

The operating modes are preselected by the operator. If Lenze’s MMI is used, the selected submodes are stored in the DB2 in DW128.

Effect in the PLC: operating mode-oriented enable or disable.

Data word	Name	Direction	Type of signal
132.00	NC program is running	NC → PLC	static

Value ”1” of the signal indicates that the processing of an NC program or a traversing block (”Manual individual block”) is active. The signal is pending until the end of the program (M02, M30), program termination (”NC program stop”) or internal termination due to an error, which cannot be remedied, or the end of a block or block termination (manual operation).

Effect in the PLC: Evaluation for disable, synchronization, time measurement.

Data word	Name	Direction	Type of signal
132.02	Block search active	NC → PLC	static

Value ”1” of the signal indicates that the processing of an NC program is active in the block search. The signal is pending until the target block of the block search has been reached.

Data word	Name	Direction	Type of signal
132.03	Individual block is running	NC → PLC	static

Value ”1” of the signal indicates that the processing of an individual block is active, triggered by the individual block function (see ETC MMI) or by input on the MMI (manual operation). The signal is pending until block termination (”NC program stop”) or internal termination due to an error which cannot be remedied.

Effect in the PLC: Evaluation for disable, synchronization, time measurement.

Data word	Name	Direction	Type of signal
0132.04	Interrupt active	PLC → NC	static

Value ”1” of the signal indicates that the NC is in the ”Interrupt” state (see also ”Interrupt” signal).

Effect in the PLC: None.

Data word	Name	Direction	Type of signal
132.08	All axes stopped	NC → PLC	static

The signal always has value ”1” when the axes have stopped or no command for traversing the axis is currently active.

The signal ”All axes stopped” is generated from the state signals of all axes.

Data word	Name	Direction	Type of signal
133.00-07	Optional hold	NC → PLC	static

This signal from the MMI activates a ”Programmed hold” through M01.

Effect in the NC:

0 = optional hold inactive
1 = optional hold active

Data word	Name	Direction	Type of signal
133.08-15	Program start counter	NC → PLC	static

This counter is incremented by 1 for every rising edge by ”NC program is running”.

Data word	Name	Direction	Type of signal
134.00-15	Axis moves, axis 0 ... 15	NC → PLC	static

The signal always has value ”1” when an axis is in motion or a command for traversing the axis is active. Therefore, the signal also has value ”1” if the movement of the axis has been temporarily interrupted by ”Feed enable” or ”override=0”. The signal is set to ”0” after the target has been reached.

Effect in the PLC: Specific disables (e.g. in manual operation) while the axes are moving.

Data word	Name	Direction	Type of signal
135.00-15	PLC axis reached target, axis 0 -... 15	NC → PLC	static

The signal has value ”1” if an axis has been started from the PLC, has reached the target and the traversing command 110 is still pending in the traverse key.

Effect in the PLC: The traverse key can be reset to zero.

Data word	Name	Direction	Type of signal
137.00-07	Program start	NC → PLC	static

The start signal is a byte information, which can assume a value between 0 and 255. A value greater than ”0” is output if the information in the virtual keypad of the NC has a value unequal to ”0”. The signal is pending for as long as the information is pending in the virtual keypad. The information is divided in two half bytes (nibbles). The nibble with the higher value holds information about the program type to be started; the one with the lower value the mode to used for start-up. The information is defined by the MMI.

Values of the nibble with the higher value

0	Start of the NC program entered in the P field (512)
1	Start of the NC program entered in the P field (514)
2	Start of the NC program entered in the P field (516)
3	Start of the NC program entered in the P field (518)
4	Start of the NC program entered in the P field (520)
5	Start of the NC program entered in the P field (522)
6	Start of the NC program entered in the P field (524)
7	Start of the NC program entered in the P field (526)
8	
9	
10	
11	
12	
13	
14	
15	Start of the automatic home position approach

Values of the nibble with the lower value

0	No response
1	Start (normal procedure)
2	Start block search (P528-P532 configuration of the entry point)
3	
4	
5	
6	
7	
8	
9	
10	
11	Resumption of a program which was interrupted by an error at a defined point (G10)
12	Resumption of an interrupted program after normal operation has been resumed (implicitly)
13	Resumption of a program which has been interrupted by an error without any previous correction measures
14	Start after M0/M1
15	Start of the next block in individual operation

Effect in the PLC: Checking the start prerequisites for the respective start mode and, if required, enable of the program start by setting the ”Programmstart” (”Program start”) signal (DW 009). Output of a corresponding error message if the start is denied.

Data word	Name	Direction	Type of signal
137.08 - 15	Individual/following block active	NC → PLC	static

Changing over from following to individual operation and vice versa. (copy from the virtual keyboard).

Effect in the NC: Processing of a program either continuously (following block) or block by block (individual block).

0 = following block; 1 = individual block

Data word	Name	Direction	Type of signal
138.00-15 139.00-15	State CAN modules (SLIO)	NC → PLC	static

Data word 138.00-15: signal state ”1” indicates that the corresponding CAN module is available.

Data word 139: A value unequal to 0 means that an error has occurred in connection with the CAN modules.

Effect in the PLC: Option of monitoring the CAN modules.

Data word	Name	Direction	Type of signal
142.00	Program stop	NC → PLC	static

Image of the ”Stop” key. The signal has value ”1” for as long as the information in the virtual keyboard of the NC has value ”1”. Value ”1” is used to request the termination of a running program or a traversing movement. The meaning depends on the set operating mode.

Manual operation: termination of the currently running movement

Automatic: termination of the currently running program

Effect in the PLC: When value ”1” is recognized, the ”Program stop” signal (DW 012.XX) is set after checking any other possible existing conditions.

Data word	Name	Direction	Type of signal
144.00	Feed stop	NC → PLC	static

Image of the ”Feed stop” keys. It has the value ”1” for as long as the information in the virtual keyboard of the NC has the value ”1”. The value ”1” requests the axes to stop without canceling the program or a traversing movement.

Effect in the PLC: If the value ”1” of the signal is detected and after any other existing conditions have been checked, the PLC withdraws the ”Feed enable” signal for all axes. Feed enable must be restored in the case of ”Program start” and ”Start after feed stop”.

Data word	Name	Direction	Type of signal
150.00	Strobe M function	NC → PLC	Strobe

The signal is set to ”0” if the data word is valid for the ”M function”. It is set to value ”1” if the acknowledgement signal of the PLC has changed from ”1” to ”0”. After switching on the control, the signal has the value ”1”.

Effect in the PLC: If the signal has the value ”0”, the data word of the ”M function” is accepted. The acknowledgement signal is set to the value ”0” after acceptance. If the strobe has value ”1”, the acknowledgement signal is set to ”1”.

In the case of synchronized M functions, the ”Read enable” signal is withdrawn. This takes place before the acknowledgement signal is set.

Data word	Name	Direction	Type of signal
151	M function	NC → PLC	Message

The data word contains the number of the M function in binary representation (0 ... 9999). The M function can be programmed in the parts program or generated by pressing the key.

The numbers of the M functions are partially defined by standards (DIN 66025); the free numbers can be assigned application-specifically. For a list of defined M-numbers for the ETC including description, refer to the ETC programming NC operating system.

Effect in the PLC: Execution of the function specified via the M function. Read enable is activated after the function has been completed or at another suitable point in time.

Data word	Name	Direction	Type of signal
159.00	Strobe H function	NC → PLC	Strobe

The signal is set to ”0” if the data word is valid for the ”H function”. It is set to value ”1” if the acknowledgement signal of the PLC has changed from ”1” to ”0”. After switching on the control, the signal has the value ”1”.

Effect in the PLC: If the signal has the value ”0”, the data word of the ”H function” is accepted. The acknowledgement signal is set to the value ”0” after acceptance. If the strobe has value ”1”, the acknowledgement signal is set to ”1”.

If the data acceptance takes longer (>10 sec.), the ”Read enable” signal must be withdrawn. This must take place before the acknowledgement signal is set.

Data word	Name	Direction	Type of signal
160	H function	NC → PLC	Message

The data word contains the number of the H function in binary representation. For the description on the acceptance of an H function, refer to the ”Programming instructions”.

Effect in the PLC: Execution of the function specified via the H function. Read enable is activated after the function has been completed or at another suitable point in time.

Data word	Name	Direction	Type of signal
177.00-15	Home position approached	NC → PLC	static

The signal has the value ”1” if a home position approach has been carried out successfully and the value ”0” after the control has been switched on and if the home position is unknown.

Effect in the PLC: Checking the home position approach and disabling machine functions or traversing movements (feed enable) if the home position is not approached.

Data word	Name	Direction	Type of signal
178.00-15	Limit switch + active	NC → PLC	static

The signal has the value ”1” if the positive limit switch of an axis is active, otherwise value ”0”.

Effect in the PLC: Disabling machine functions or traversing movements (feed enable).

Data word	Name	Direction	Type of signal
179.00-15	Limit switch - active	NC → PLC	static

The signal has the value ”1” if the negative limit switch of an axis is active, otherwise value ”0”.

Effect in the PLC: Disabling machine functions or traversing movements (feed enable).

Data word	Name	Direction	Type of signal
180.00-15	Reference switch active	NC → PLC	static

The signal has the value ”1” if the reference switch of an axis is active, otherwise value ”0”.

Effect in the PLC: If required, man. execution of homing via the PLC.

Data word	Name	Direction	Type of signal
182.00-15	Slave axis limit switch + active	NC → PLC	static

The signal has the value ”1” if the positive limit switch of the slave axis of a synchronous axis is active, otherwise the value is ”0”.

Effect in the PLC: Disabling machine functions or traversing movements (feed enable).

Data word	Name	Direction	Type of signal
183.00-15	Slave axis limit switch - active	NC → PLC	static

The signal has the value ”1” if the negative limit switch of the slave axis of a synchronous axis is active, otherwise the value is ”0”.

Effect in the PLC: Disabling machine functions or traversing movements (feed enable).

Data word	Name	Direction	Type of signal
184.00-15	Slave axis reference switch active	NC → PLC	static

The signal has the value ”1” if the reference switch of the slave axis of a synchronous axis is active, otherwise the value is ”0”.

Effect in the PLC: If required, manual execution of homing via the PLC.

Data word	Name	Direction	Type of signal
192.00-199.15	State CAN modules (CANopen)	NC → PLC	static

Signal state ”1” indicates that the corresponding CANopen module is available. The bits in the data words are assigned by means of the module ID (node no.).

Example:

192.01 indicates state of node 1
192.15 indicates state of node 15
193.01 indicates state of node 17

Effect in the PLC: Option of monitoring the CAN modules.

Data word	Name	Direction	Type of signal
200	General override for axes	NC → PLC	static
201	Override for spindle speed	NC → PLC	static
202	Override for oscillation speed	NC → PLC	static
203	Override for PLC axes (Target position approach)	NC → PLC	static

Evaluation factor which is set on the HMI for the currently valid traversing speed. The specification is made in steps of 0.1 % (copy from the virtual keyboard).

Effect in the NC: The selected value is used as a factor for evaluating the currently programmed maximum speed of the rough interpolator. The override is effective both in manual as well as in automatic operation. It can be inhibited, for example, by M functions within a program.

Data word	Name	Direction	Type of signal
204.00-07	Traverse key axis 0	HMI → PLC	static
204.08-15	Traverse key axis 1	HMI → PLC	static
...	...	...	...
...	...	...	....
212.00-07	Traverse key axis 14	HMI → PLC	static
212.08-15	Traverse key axis 15	HMI → PLC	static

Like description for DW 84-91. Only significant if traverse key enable of the PLC is activated and if traverse keys are available for the MMI.

Data word	Name	Direction	Type of signal
212.00-15 213.00-15 214.00-15 215.00-15 216.00-15 217.00-15 218.00-15 219.00-15	Area of the ”Virtual keyboard”	HMI → PLC	static

Area of the virtual keyboard whose meaning/function can be freely defined. If the ETC-MMI (PC user interface) is used, this is where the states of the freely configurable PLC keys (soft keys) are stored bit by bit (see chapter ”ETC-MMI”).

5.1.3 Data block 2

The communication between MMI and PLC takes place via this data block. An image of the data block is provided in the Dual-Port-RAM.

Data word 0 ... 127 contains signals from the PLC to the MMI. Data word 128 ... 191 contains signals from HMI to the PLC.

From data word 224, also all data of the machine constant MK_DW224_255, which may be used application-specifically, is stored. This is the data for configuring the machine.

The distribution of the data is left to the user and must be specifically defined. The NC only provides for the data exchange.

The machine constants are copied into the data block when the system software is loaded and if there is a permissible change of the machine constants. The data is taken from the general machine constants for the configuration of the machine. There is a reserved area for the configuration of the PLC (see chapter ”Machine constants”).

Assignment

For the ETC-MMI, the assignment of the DB2 listed in the table below is used.

This assignment is a recommendation for the use of the DB2. Within the areas data word 0 ... 127 and 128 ... 191 a freely definable assignment is possible.

Data word	Name	Direction
000.00 – 007.15	States (28 bits)	PLC → HMI
008.00 – 015.15	Key control (128 bits)	PLC → HMI
016.00 – 079.15	Displays (64 data word)	PLC → HMI
080.00 – 095.15	Notes/static errors (256 bits)	PLC → HMI
096.00 – 096.15	Message buffer: acknowledgement counter	PLC → HMI
097.00 – 097.15	Message buffer: counter 1	PLC → HMI
098.00 – 125.15	Message buffer: user data (28 data words)	PLC → HMI
126.00 – 126.15	Message buffer: counter 2	PLC → HMI
127.00 – 127.15	Monitoring of PLC is running	PLC → HMI
128.00 – 128.15	Sub mode	PLC ← HMI
129.00 – 143.15	States (256 bits)	PLC ← HMI
144.00 – 159.15	Keys (256 bits)	PLC ← HMI
160.00 – 160.15	Message buffer: acknowledgement counter	PLC ← HMI
161.00 – 161.15	Message buffer: counter 1	PLC ← HMI
162.00 – 189.15	Message buffer: user data (28 data word)	PLC ← HMI
190.00 – 190.15	Message buffer: counter 2	PLC ← HMI
191.00 – 191.15	Monitoring of HMI is running	PLC ← HMI
224.00 – 255.15	Freely available machine constants	PLC ← NC

If the key control is activated (DB1.31.4), the 128 bits from DB2.8.0 to DB2.15.15 correspond to the MMI keys DB1.212.0 to DB1.219.15.

Explanation

DW080.00-095.15	Signal state ”1” has the effect that a text is displayed in the PLC messages window on the MMI. The texts are stored in a file (default:Lenze.txt). DW080.00 is linked with the entry 01020001, DW080.01 with the entry 01020002 etc.
DW128	If the ETC-MMI is used, the selected sub mode is stored here: 0: no sub mode selected; operating mode (DW131) = 1 **Operating mode (DW131) = 1** 1: step travel 2: modal traveling 3: target point traveling 4: handwheel 5: manual homing 6: automatic homing 7: zero points **Operating mode (DW131) = 3** 1: block search **Operating mode (DW131) = 4** 1: teaching
DW127.00-127.15, DW191.00-191.15	The data words are used for mutual control of PLC and MMI. This way, they can monitor whether the counterpart is still ”alive”. Operating mode: PLC: DW127 := DW191; HMI: DW191 := DW127 + 1; If the data words do not differ over a certain period, the PLC can detect that the MMI is no longer ”alive”.
DW224.00-255.15	The values of the machine constants MK_DW224_255 are stored one-to-one in these data words.

5.2 Extended interface for MMI functions

5.2.1 Data blocks 8 ... 14

Assignment

DB8 data word	Name	Direction
DW000 - 255	reserved	

DB9 data word	Name	Direction
DW000 - 255	reserved	

DB10 data word	Name	Direction
DW000 - 255	reserved	

DB11 data word	Name	Direction
DW000 - 247	reserved	
DW250 - 253	Target position block axis 0	PLC ← NC
DW254 - 255	Target position block axis 1 (LOW-WORD)	PLC ← NC

DB12 data word	Name	Direction
DW000 - 255	reserved	
DW000 - 001	Target position block axis 1 (HIGH-WORD)	PLC ← NC
SW002 - 005	Target position block axis 2	PLC ← NC
DW006 - 009	Target position block axis 3	PLC ← NC
DW010 - 013	Target position block axis 4	PLC ← NC
DW014 - 017	Target position block axis 5	PLC ← NC
DW018 - 021	Target position block axis 6	PLC ← NC
DW022 - 025	Target position block axis 7	PLC ← NC
DW026 - 029	Target position block axis 8	PLC ← NC
DW030 - 033	Target position block axis 9	PLC ← NC
DW034 - 037	Target position block axis 10	PLC ← NC
DW038 - 041	Target position block axis 11	PLC ← NC
DW042 - 045	Distance between current zero point and home position axis 0	PLC ← NC
DW046 - 049	Distance between current zero point and home position axis 1	PLC ← NC
DW050 - 053	Distance between current zero point and home position axis 2	PLC ← NC
DW054 - 057	Distance between current zero point and home position axis 3	PLC ← NC
DW058 - 061	Distance between current zero point and home position axis 4	PLC ← NC
DW062 - 065	Distance between current zero point and home position axis 5	PLC ← NC
DW066 - 069	Distance between current zero point and home position axis 6	PLC ← NC
DW070 - 072	Distance between current zero point and home position axis 7	PLC ← NC
DW074 - 077	Distance between current zero point and home position axis 8	PLC ← NC
DW078 - 081	Distance between current zero point and home position axis 9	PLC ← NC
DW082 - 085	Distance between current zero point and home position axis 10	PLC ← NC

DB12 data word	Name	Direction
DW086 - 089	Distance between current zero point and home position axis 11	PLC ← NC
DW090 - 093	Actual position block axis 0	PLC ← NC
DW094 - 097	Actual position block axis 1	PLC ← NC
DW098 - 101	Actual position block axis 2	PLC ← NC
DW102 - 105	Actual position block axis 3	PLC ← NC
DW106 - 109	Actual position block axis 4	PLC ← NC
DW110 - 113	Actual position block axis 5	PLC ← NC
DW114 - 117	Actual position block axis 6	PLC ← NC
DW118 - 121	Actual position block axis 7	PLC ← NC
DW122 - 125	Actual position block axis 8	PLC ← NC
DW126- 129	Actual position block axis 9	PLC ← NC
DW130 - 133	Actual position block axis 10	PLC ← NC
DW134 - 137	Actual position block axis 11	PLC ← NC
DW137	Following error axis 0 in increments	PLC ← NC
DW139	Following error axis 1 in increments	PLC ← NC
DW140	Following error axis 2 in increments	PLC ← NC
DW141	Following error axis 3 in increments	PLC ← NC
DW142	Following error axis 4 in increments	PLC ← NC
DW143	Following error axis 5 in increments	PLC ← NC
DW144	Following error axis 6 in increments	PLC ← NC
DW145	Following error axis 7 in increments	PLC ← NC
DW146	Following error axis 8 in increments	PLC ← NC
DW147	Following error axis 9 in increments	PLC ← NC
DW148	Following error axis 10 in increments	PLC ← NC
DW149	Following error axis 11 in increments	PLC ← NC
DW150 - 151	Speed of spindle 0	PLC ← NC
DW152 - 153	Speed of spindle 1	PLC ← NC
DW154 - 155	Speed of spindle 2	PLC ← NC
DW156 - 157	Speed of spindle 3	PLC ← NC
DW158 - 159	Speed of spindle 4	PLC ← NC
DW160 - 161	Speed of spindle 5	PLC ← NC
DW174 - 175	Event counter	PLC ← NC
DW176 - 177	Event interval	PLC ← NC
DR178	Operating mode	PLC ← NC
DL178	Program processing active	PLC ← NC
DR179	Block search	PLC ← NC
DL179	Measuring system (mm/inch)	PLC ← NC
DR180	Hide block	PLC ← NC
DL180	Individual block	PLC ← NC
DR181	Optional hold	PLC ← NC
DL181	Programmed stop	PLC ← NC
182.00 - 182.15	Home position approached yes/no	PLC ← NC
183.00 - 183.15	Limit switch + approached yes/no	PLC ← NC
184.00 - 184.15	Limit switch – approached yes/no	PLC ← NC
185.00 - 185.15	Reference cams approached yes/no	PLC ← NC
186.00 - 186.00	Reserve input (AR)	PLC ← NC
DW194 - 197	Angle basic rotation	PLC ← NC

DB12 data word	Name	Direction
DW198 - 201	Following error of axis 0 in input units	PLC ← NC
DW202 - 205	Following error of axis 1 n input units	PLC ← NC
DW206 - 209	Following error of axis 2 in input units	PLC ← NC
DW210 - 213	Following error of axis 3 in input units	PLC ← NC
DW214 - 217	Following error of axis 4 in input units	PLC ← NC
DW218 - 221	Following error of axis 5 in input units	PLC ← NC
DW222 - 225	Following error of axis 6 in input units	PLC ← NC
DW226 - 229	Following error of axis 7 in input units	PLC ← NC
DW230 - 233	Following error of axis 8 in input units	PLC ← NC
DW234 - 237	Following error of axis 9 in input units	PLC ← NC
DW238 - 241	Following error of axis 10 in input units	PLC ← NC
DW242 - 245	Following error of axis 11 in input units	PLC ← NC
DW254	Override axes	PLC ← NC
DW255	Override spindles	PLC ← NC

DW174 - 175: Event counter; number of positive edges on the ”Reserveeingang der ersten Achse” (”Reserve input of the first axis”).

DW176 - 177: Event interval; interval between two events (see DW174 - 175) in microseconds. Lowest resolution: fine interpolation cycle.

DB13 data word	Name	Direction
DR034	reserved (normally channel number)	PLC ← NC
DL034	Program active	PLC ← NC
DR035	Current tool number	PLC ← NC
DL035	Current workpiece number	PLC ← NC
DW036 - 037	Active identification axis 0 - 3	PLC ← NC
DW038 - 041	Band speed	PLC ← NC
DW042	Current program number	PLC ← NC
DW043	Current programmed block number	PLC ← NC
DW044	Current relative block number	PLC ← NC
DW045	Last M function	PLC ← NC
DW046	Current dwell time	PLC ← NC
DR049	Level	PLC ← NC
DL049	Path / route operation	PLC ← NC

DB14 data word	Name	Direction
DW000 - 255	reserved	

5.2.2 Data block 15

Assignment

The data block 15 represents the ”virtual keyboard” of the NC computer. This block can only be written by the PLC if no ETC-MMI is connected to the control. Otherwise the contents of the PLC can only be read.

Data word	Name	Direction
DR010	Optional hold	HMI → NC
DL010	Individual/following block	HMI → NC
DR011	Hide block	HMI → NC
DW028 - 033	Traverse keys axis 0 - 11	HMI → NC
DW036 - 039	Override selector switch 1 - 4	HMI → NC
DW040 - 043	Speed preselection for traverse keys	HMI → NC
DW044 - 047	Increment preselection	HMI → NC

Description

Data word	Name	Direction	Type of signal
010.00-07	Optional hold	HMI → NC	static

Activation of the execution of M01. This signal is not evaluated by the NC core, the corresponding function must be implemented in the PLC, if required, by withdrawing the feed or read enable for M01.

Effect in the NC:

0 = optional hold inactive
1 = optional hold active

Data word	Name	Direction	Type of signal
010.08-15	Individual/following block	HMI → NC	static

Change over from following to individual block operation and vice versa.

Effect in the NC: Processing of a program either continuously (following block) or block by block (individual block).

0 = following block
1 = individual block

Data word	Name	Direction	Type of signal
011.00-07	Hide block	HMI → NC	static

With this signal, the ”Satz ausblenden” (”Hide block”) function can be switched on or off.

Effect in the NC: Blocks with a preceding ”/” are hidden, i.e. they are not executed:

0 = Hide block inactive
1 = Hide block active

Data word	Name	Direction	Type of signal
028.00-07	Traverse key axis 0	HMI → NC	static
028.08-15	Traverse key axis 1	HMI → NC	static
029.00-07	Traverse key axis 2	HMI → NC	static
029.08-15	Traverse key axis 3	HMI → NC	static
030.00-07	Traverse key axis 4	HMI → NC	static
030.08-15	Traverse key axis 5	HMI → NC	static
031.00-07	Traverse key axis 6	HMI → NC	static
031.08-15	Traverse key axis 7	HMI → NC	static
032.00-07	Traverse key axis 8	HMI → NC	static
032.08-15	Traverse key axis 9	HMI → NC	static
033.00-07	Traverse key axis 10	HMI → NC	static
033.08-15	Traverse key axis 11	HMI → NC	static

These signals are used for traversing the axes. One byte is available for each axis. By writing the bytes, the respective action is carried out.

Effect in the NC:

-100 to +100: traversing with % of the maximum speed. The sign defines the direction.

+/-101: inching by increment (corresponds to DW44 ... 47) on the path if there is no feed enable. Only available in the case of a programmed traversing movement.

+/-102: home position approach (sign is ignored).

+/-103: setting zero point (sign is ignored)

+/-104: step travel.

+/-105: handwheel active - evaluation factor 1.

+/-106: handwheel active - evaluation factor 3.

+/-107: handwheel active - evaluation factor 10.

+/-108: handwheel active - evaluation factor 30.

+/-109: handwheel active - evaluation factor 100.

+/-110: traverse command for PLC axis

+/-111: traversing with the speed from P208 ... 223.

The sign defines the direction. (📖 136).

Data word	Name	Direction	Type of signal
036	General override for axes	HMI → NC	static
037	Override for spindle speeds	HMI → NC	static
038	Override for oscillation speed	HMI → NC	static
039	Override for PLC axes (Target position approach)	HMI → NC	static

Definition of a signed evaluation factor for the currently valid traversing speed. The specification is made in steps of 0.1 %.

Effect in the NC: The selected value is used as factor for evaluating the currently programmed maximum speed by the rough interpolator.

The override is effective in both manual as well as automatic operation. It can be inhibited within a program via M functions.

Data word	Name	Direction	Type of signal
040-043	Speed preselection for traverse keys	HMI → NC	static

Definition of speed during modal traveling using the traverse keys. Specification in [mm/min] as 64-bit floating point number.

Effect in the NC: The selected value is accepted at a fixed position in the parameter field. Every time the traverse key is actuated, the respective axis is moved modally at the speed entered.

Data word	Name	Direction	Type of signal
044-047	Increment preselection	HMI → NC	static

Definition of increment during manual traversing in inching mode. Specification in [mm] or [inch] as 64-Bit floating point number.

Effect in the NC: The selected value is accepted at a fixed position in the parameter field. Every time the traverse key is actuated, the respective axis is moved about the delta specified.

6 ET -MMI gateway

The MMI gateway is the communications program between Windows applications and ETC control systems. Different applications such as MMIs, configuration tools or OPC servers can establish connections to one or more control systems at the same time.

The gateway implements all required mechanisms for access control, error handling and diagnostics and supports control-specific hardware drivers and communication protocols. By means of the configuration tools, new connections can be added and existing connections can be edited.

6.1 Installing the ETC-MMI gateway

The ETC-MMI Gateway is installed during the installation of the Lenze ETC-MMIs. It can also be installed as a separate application. In this case, it is available as communication channel for other applications.

During installation, the following files are copied to the system directory on the PC (e.g. c:\Windows\System32):

File	Description
mmigtway.exe	Gateway program
mmictr.dll	Interface for the application
ipcom.dll	DLL with internal gateway functions
gtwconf.exe	Configuration interface
mmigtwayini	Configuration file 📖 283

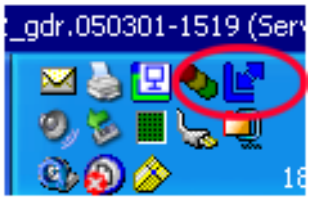
Note!

If you want to install the ETC-MMI gateway without ETC-MMI, manually copy the files to your system directory.

6.2 Starting the ETC-MMI gateway

The ETC-MMI gateway is started automatically when an application (e.g. ETC-MMI) loads the file ”mmictrl.dll”. The current configuration is read from the file ”mmigtway.ini” (📖 283) and checked.

In the task bar, an icon for the ETC-MMI gateway is displayed. You can open the menu with a mouse-click on the gateway icon.



ETCN001

ETCN002

Settings: Start configuration interface.

About: Display version and manufacturer information.

Exit: Close gateway (if there are active connections to an application, a warning is displayed).

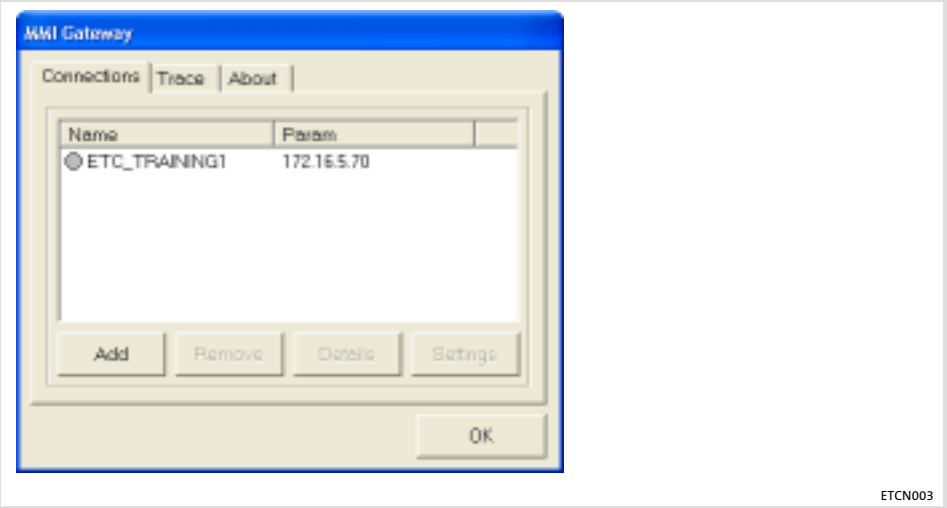
6.3 Configuring the ETC-MMI gateway

Via the configuration interface of the ETC-MMI gateway, you can configure the connections to the control systems and call debug information.

The configuration interface is a separate application (gtwconf.exe), which is installed with the ETC-MMI gateway. It can be started via the gateway menu (📖 277) or as Windows application, e.g. via Windows File Explorer.

6.3.1 Connection - Setting up connections

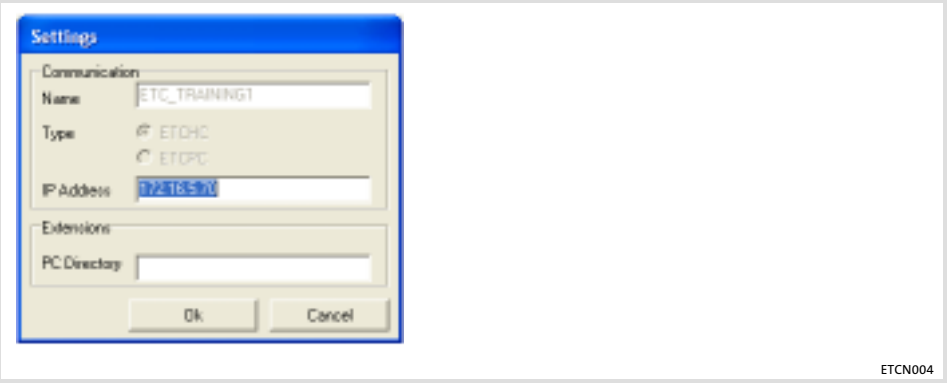
The ”Connections” tab shows the list of the configured connections.



A green LED next to the connection name indicates an active connection, via which the messages and/or cyclic data are transferred.

Add - create new connection

Use the **Add** button to create a new connection. In the ”Settings” dialogue, you determine the communication parameters.



Name: To enable an application to communicate with a control system via the ETC-MMI gateway, each connection must be assigned an unambiguous name. You can choose any name. Assign e.g. consistent names ”ETC0”, ”ETC1” etc. or application-specific names ”ramp”, ”laser control” etc. A name can contain up to 31 ASCII characters (”A ... Z”, ”a ... z” , ”0 ... 9”).

Type: Determine the connection type:

- ► UDP connection for ETCHx control systems
  Ethernet communication via DPR/UDP/IP protocol.
- ► PCI connection for ETCPx control systems
  DPR communication via a WDM driver (etcpc.sys, in the scope of supply of the ETCPC).

IP address: Specify the communication parameters that depend on the connection type:

- ► ETCHC = IP address (🕮 28)
- ► ETCPC = Index of the PCI cards (indexes of the PCI cards are in the range 0 ... 9 and are automatically assigned by the driver. The first ETCPC has the index 0).

PC Directory:

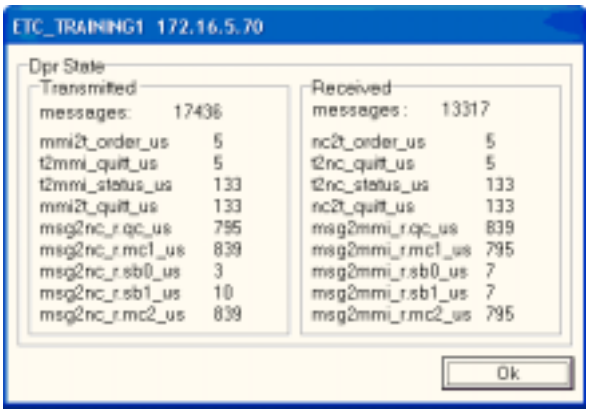
<Ok>: Use this button to write entries in the file ”mmigtway.ini” (🕮 283). If a connection with the same parameters already exists, the error ”Gateway cannot change the parameter” is reported.

Remove – Delete connection

Use the **Remove** button to delete the selected connection. You can only delete connections that are not used by any application.

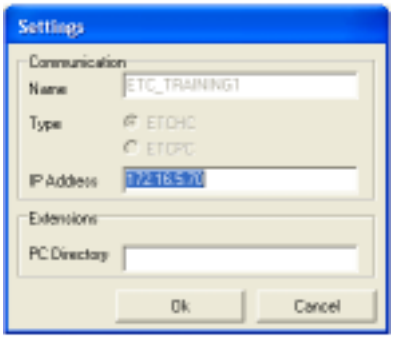
Details – Display communication status

If you want to display the communication status of the selected connection, click the **Details** button.



Settings – Edit connection parameters

Via the **Settings** button, you can edit the communication parameters of the selected connection if e.g. the IP address of the control system has changed.



Name and **Control** cannot be changed. If you also want to change the control type, delete a connection and create a new connection with the same name.

IP address: Specify the communication parameters that depend on the control type:

- ETCHC = IP address (📖 28)
- ETCPC = Index of the PCI cards (indexes of the PCI cards are in the range 0 ... 9 and are automatically assigned by the driver. The first ETCPC has the index 0).

<Ok>: Use this button to write entries in the file ”mmigtway.ini” (📖 283). If a connection with the same parameters already exists, the error ”Gateway cannot change the parameter” is reported.

New parameters are adopted immediately by the gateway.

6.3.2 Trace – Error logbook

In the case of faults in the communication, you can activate trace logs on this tab. Gateway traces are internal events of the communication channel between the application and the gateway and contain information on the communication flow or the causes of occurred errors. Trace logs are saved in the file ”mmigtway.trc”, which is created anew every time the gateway runs up.

Note!
We recommend to deactivate all trace logs in normal operation.

Select the respective control fields to record trace information for the following events:

OnChannelCreate: Create a new communication channel in the gateway.

OnConnect: Open/close a communication channel.

OnRead: Receive a message via the communication channel.

OnWrite: Send a message via the communication channel.

OnLoadFirmware: Transfer the firmware.

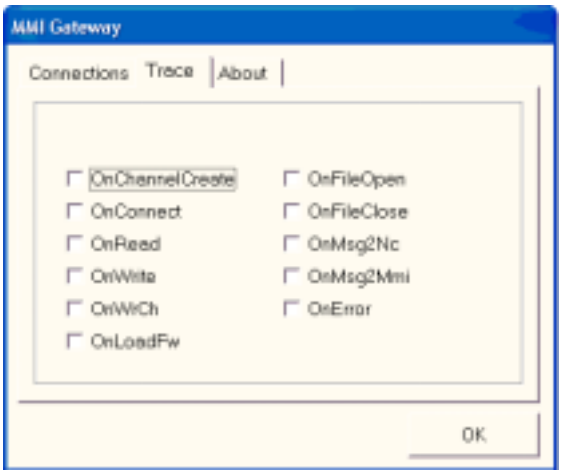
OnFileOpen: Open a file.

OnFileClose: Close a file.

OnMsg2Nc: Send a message to the control system.

OnMsg2Mmi: Send a message to the application (ETC-MMI).

OnError: If an error occurs.

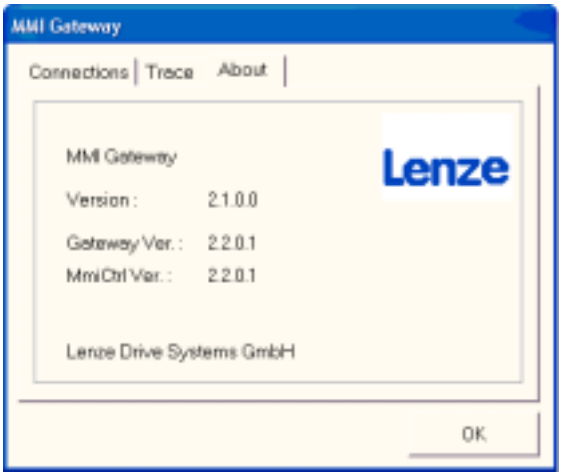


ETCN061

Activated traces are automatically switched on the next time the gateway is started.

6.3.3 About – Version information

The ”About” tab shows the version numbers of the gateway, the configuration tool and the MmiCtrl.dll.



ETCN062

6.4 Mmigtway.ini

In the file ”mmigtway.ini”, the current configuration of the gateway is saved.

[Connection] section

In the [Connections] sections, general information of the gateway is saved. The following key variables are defined:

- ► UdpConns = number of UDP connections (number of ETCHCs)
- ► PciConns = number of PCI connections (number of ETCPCs)
- ► DemoConns = reserved for simulated connections

[Options] section

In the [Options] section with the key variable ”AutoClose”, you can determine whether the gateway is to be closed automatically when there is no connection to an application.

[Traces] section

The key variables defined in the [Traces] section are the active traces. If the value of the variable is non-zero, the corresponding trace information is logged in the file ”mmigtway.trc”. (🕮 281).

Sections of the connections

For each connection, another section is created. ”*” is the index of the connection:

- ► [UdpConn*] - for UDP connection (ETCHC)
- ► [PCIConn*] - for PCI connection (ETCPC)
- ► [DemoConn*] - for demo connection

The names of the connection, the communication parameter (IP address for UDP connections, index for PCI cards) and the PC disk path are saved under the key variables ”Name”, ”Param”, ”PcDir”.

6.4.1 Example of the file ”mmigtway.ini”

```
[Connections]
UdpConns=3
PciConns=1
DemoConns=0
[Options]
AutoClose=0
[Traces]
OnChCreate=0
OnConnect=0
OnRead=0
OnWrite=0
OnWrCh=0
OnLoadFw=0
OnFileOpen=0
OnFileClose=0
OnMsg2Nc=0
OnMsg2Mmi=0
OnError=0
ConnTrc=0
OnAddConn=0
[UdpConn0]
Name=CNC0
Param=172.16.5.113
PcDir=
[UdpConn1]
Name=CNC1
Param=172.16.5.114
PcDir=
[PciConn1]
Name=MyPnc
Param=0
PcDir=
[UdpConn2]
Name=CNC2
Param=172.16.5.115
PcDir=
```

6.5 Communication values in the DPR area

Name	Meaning
Communication area MMI->NC	
mmi2t_order_us	Command to the communication processor
t2mmi_quitt_us	Acknowledgement from the communication processor
t2mmi_status_us	Status from the communication processor
mmi2t_quitt_us	Acknowledgement to the communication processor
msq2nc_r.qc_uc	Message acknowledgement counter
msq2nc_r.mc1_us	Messages-Start-Message counter
msq2nc_r.sb0_us	Messages-Main group
msq2nc_r.sb1_us	Messages-subgroup
msq2nc_r.mc2_us	Messages-End-Message counter
Communication area NC-> MMI	
nc2t_order_us	Command to the communication processor
t2nc_quitt_us	Acknowledgement from the communication processor
t2nc_status_us	Status from the communication processor
nc2t_quitt_us	Acknowledgement to the communication processor
msq2mmi_r.qc_uc	Message acknowledgement counter
msq2mmi_r.mc1_us	Messages-Start-Message counter
msq2mmi_r.sb0_us	Messages-Main group
msq2mmi_r.sb1_us	Messages-subgroup
msq2mmi_r.mc2_us	Messages-End-Message counter

7 ETC-MMI

The program ”ETC-MMI” is used for the following tasks:

- ► Configuring the control system
- ► Operating and monitoring the control system
- ► Maintenance of the control system and error diagnosis

7.1 Installing ETC-MMI

Stop!

Only install the PCI control variant ETCPx after installing the ETC-MMI and before starting the ETC-MMIs.

Note!

The ETC-MMI Gateway is installed during the installation of the Lenze ETC-MMIs

1. In Windows File Explorer, open the program ”setup.exe” on the ETC-MMI installation CD.
2. Follow the instructions of the installation program. The following will be requested:
   – Demo or standard installation
   – Control type (ETCHC or ETCPC)
   – IP address of the ETCHC (📖 28) or index of the ETCPC (indexes of the PCI cards are in the range 0 ... 9 and are automatically assigned by the driver. The first ETCPx has the index 0).

   In the start bar, the entry ”Lenze ETC-MMI” is created. It can be used to start the application.
3. For an ETCHC: Check that the TCP/IP protocols are installed in the network properties (**Start → Settings → Control Panel → Network**).

   For an ETCPC: Install the ETCPC plug-in card in the PC. Observe the notes in the ETC Hardware Manual.

Installed files

After a standard installation, the following files and file paths can be found on the hard disk of the PC after a successful installation:

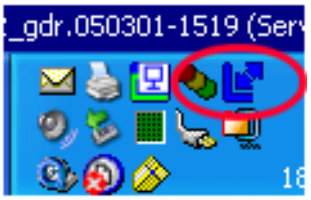
e.g. c:\Programs\Lenze\ETC\MMI\)	
Lenze.exe	ETC-MMI application
ncform.hlp	Help file
install.hlp	Installation instructions
...\cfg	Configuration directory
muster.mk	Machine constant file with basic settings
mk.hlp	ASCII file, containing help information on setting the machine constants (”edit MC”)
mmi_fehl.db	ASCII file containing error messages of the MMI
ncr_fehl.db	ASCII file containing error messages of the NC
Lenze.txt	ASCII file with general texts, notes, error messages, information on inputs/outputs and function keys. The file is required by the user interface.
sps_fehl.db	ASCII file containing error messages of the PLC
delphmmi.ini	Initialisation or configuration file for ETC-MMI
...\bin	
etc*.rsc	ETCHC or ETCPC firmware
spsdummy.prg	Example of a PLC program
...\prg	DIN file index
nikolaus.din	Example of a DIN file (profile)
...\log	Folder for all temporary files which are loaded back from the control system
e.g. c:\Windows\Systems32	
mmigrp32.dll	MMI-DLL
mmigtway.exe	Gateway program
mmictr.dll	Interface for the application
ipcom.dll	DLL with internal gateway functions
gtwconf.exe	Configuration interface
mmigtwayini	Configuration file 📖 283

7.2 Starting ETC-MMI

1. Start the ETC-MMI via **<Start> → Programs → Lenze → ETC**.

ETCN011

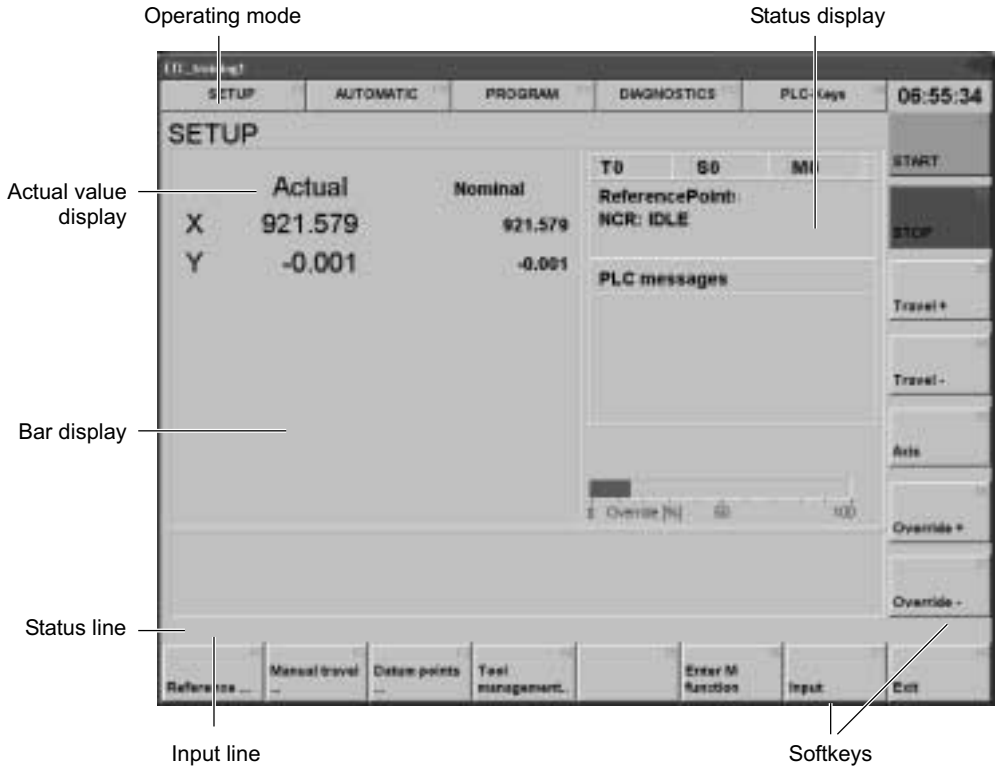
The ETC-MMI Gateway is automatically started. The application can be seen on the task bar:



ETCN001

7.3 Operating ETC-MMI

7.3.1 Display elements of the program interface



Operating modes: Displays the currently selected operating mode (”Setup”, ”Automatic”, ”Programming” or ”Diagnostics”) and, if selected, a sub mode.

Note!

It can be defined which operating mode is displayed when the user interface is started. (📖 327).

Actual value display: This area shows the actual and desired positions of the configured axes.

Status display: The status fields display current program statuses, such as the current program, T, S, and M number. Moreover, information on the currently edited parts program and PLC error messages are displayed here.

Softkeys: The horizontal and vertical softkeys are located at the lower and right window edge. Their labelling and function depends on the selected operating mode or sub mode.

Input line: The input line is used for special inputs. If the line is active, it is displayed in white.

Status line: Notes and general statuses are displayed here.

Bar display: In addition to the standard bar display for ”Override” and ”V_{actual}”, additional displays can be configured.

7.3.2 Operational controls of the program interface

The standard user interface does not require an external machine control panel for the operation of the machine. All important functions (start, stop, traverse buttons) are assigned to softkeys (function keys); i.e. all operational controls of the program interface can be activated via the keyboard.

Note!

Following the properties of a touch screen, an operation via pointer devices (e.g. mouse) is possible. However, the action must be triggered with the **<Enter>** key. In general, it is not possible to trigger actions via a ”double-click”.

If the internal PLC is programmed accordingly, the machine functions can be assigned to different keys to evaluate them in parallel or alternatively to the softkeys.

Stop!

Triggering machine movements via softkeys is not permissible for all machines. The operator must ensure that the valid safety guidelines are observed.

Operating mode-independent operational controls

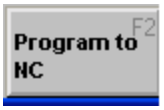
At the upper window edge, you can change the operating mode of the ETC-MMIs via the softkeys **Setup**, **Automatic**, **Program** or **Diagnostics** or via the keys **<F9>**... **<F12>**. Via the softkey **PLC keys** or via **<Shift>+<F8>**, you can switch the softkey bar to ”PLC keys”. These must be programmed in the PLC. (📖 75).

Operating mode-dependent operational controls

A description of the operating mode-dependent softkeys can be found in the descriptions of the operating modes later in this chapter. In the following, the different behaviour of softkeys when they are pressed will be explained.

Direct triggering of a function (pushbutton)

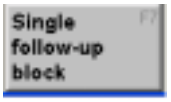
The assigned function is executed when the key is pressed or released or as long as the key is being pressed.



ETCN065

Activation of a certain status (switch)

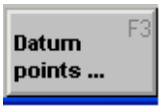
After the key has been pressed, the assigned function is executed until the same key is pressed again. As long as the function is active, the key is shown as ”pressed”. In the following, the active status is called sub mode and is shown in the operating mode line.



ETCN067

Showing a new softkey line (submenu)

These function keys open a submenu; i.e. they change the labelling and thus the meaning of other keys. The labelling of this function key type always ends with three dots.



ETCN066

Use the **<Back>** function key to change back to the calling key level.

7.3.3 Help function

Use the shortcut **<CTRL>+<SHIFT>+<any F key>**to start the online help.

7.3.4 Configuration file

The configuration file ”delphmmi.ini” contains settings which are required for operating the MMIs. It may only be changed by trained personnel. Further information (📖 327).

7.3.5 Language switch

All texts of the user interface are saved in an ASCII file, which can be changed with any editor. This way, the texts can be individually adapted or translated into foreign languages (if the characters can be displayed). The language file is set in the configuration file ”delphmmi.ini”. Further information (📖 321).

7.3.6 Passwords

In the ETC-MMI, a password can be assigned for each operating mode and additionally for changing the passwords. These passwords are prompted for when the ETC-MMIs are started or when the operating mode is changed. The following passwords are preset:

Operating mode / password administration	Password
Setup	1
Automatic	2
Program	3
Diagnostics	4
Edit password	5

Passwords can be changed in the ”Diagnostics” operating mode. (📖 312). They are saved in encrypted form in ”Lenze.pwd”.

Note!

If you delete the file ”Lenze.pwd” in the ”cfg” folder, all passwords are reset to the standards listed in the table. If you do not want any passwords to be prompted (e.g. during commissioning), enter a ”space” as password.

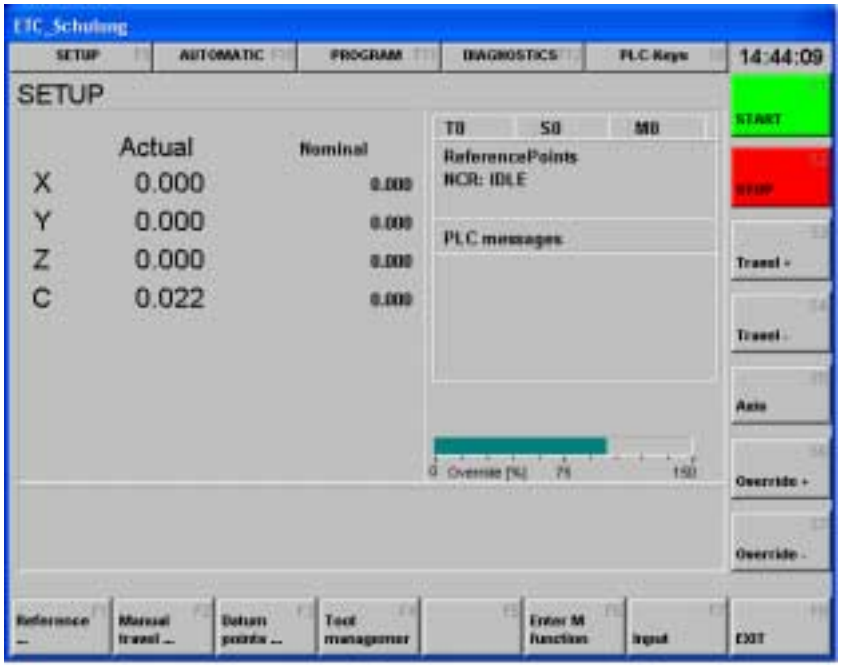
7.3.7 Notes, warnings, error messages

Notes, warnings and errors reported in the NC system are shown in a separate window. The title line displays the name of the control system reporting the error.

ETCN064

Such a window must be acknowledged with **<Enter>** before other keyboard entries are accepted (modal dialogue). The message types are differentiated by their colours. A blue background indicates a note, a yellow background a warning and a red background an error. All messages are saved in a log book file with date and time of the event on the PC and can be accessed in the ”Diagnostics” operating mode. (📖 312).

7.4 ”Setup” operating mode



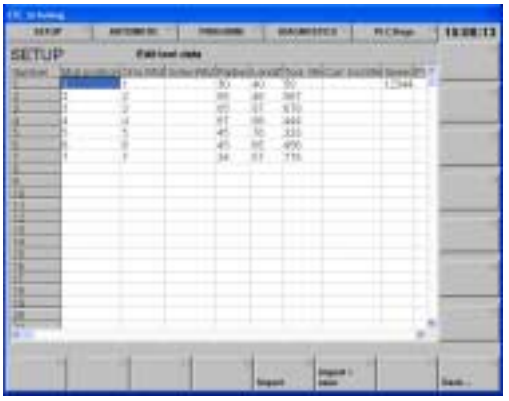
ETCN068

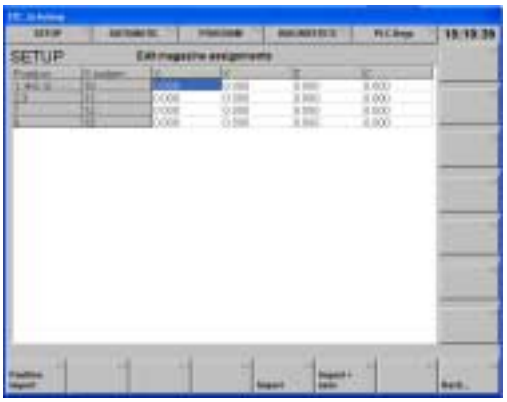
The ”Setup” operating mode contains functions for setting up the plant. Among other things, you can carry out homing and manual travel and manage tools.

Horizontal function keys

Reference...	
automatic	Via the **<Start>** key, an automatic home position approach can be initiated. All axes are referenced in the configured sequence.
manual	A manual home position approach can be initiated. You can select the desired axis by means of the **<Axis>** key (or <Cursor> keys) and traverse the axes with the **<Travel +>**/**<Travel ->** or **<Start>** key.
back...	Back to the previous level.
Manual travel...	Here you can manually traverse the axes.
Modal travel	The axis selected with the **<Axis>** key (or <Cursor> keys) is traversed as long as the **<Travel +>**/**<Travel ->** key is pressed. The speed at which the axes are traversed is determined by MK_MODVMAX. It can be influenced via the **<Override +>**/**<Override ->** key.
Step travel	The selected axis is traversed by the specified increment (see below). The direction is determined by the **<Travel +>** / **<Travel ->** keys. During ”step travel”, the axes are traversed at high rate.
Increment	Increment in mm for traversing the selected axis. The entry must be completed with **<ENTER>**. The function key is only labelled and enabled if the **<Step travel>** key has been pressed.
Target point travel	The ”Target point travel” function refers to all axes. If you press the **<Start>** key, all axes whose actual and set position is not identical are traversed. During ”Target point travel” the axes are traversed at high rate.

Target position	Target position for the selected axis. Select the desired axis by means of the **<Axis>** key (or <Cursor>keys). The entry must be completed with **<ENTER>**. **<ESC>** cancels the entry and restores the old positions. The function key is only labelled and enabled if the **<Target point travel>** key has been pressed.
Handwheel	**The “handwheel keys“ are only displayed if a handwheel is configured in the machine constants.** Traversing the axes via a handwheel. You can select the axis which you want to traverse by means of the handwheel via the **<Axis>** key (or <Cursor> keys).
Handwheel factor	Selection of a factor which acts on the traversing speed. 5 handwheel factors are provided (1, 3, 10, 30, 100). Every time the **<Handwheel factor>** key is pressed, the following factor is selected. The function key is only labelled and enabled if the **<Handwheel>** key has been pressed.
back...	Back to the previous level.
Datum points...	
Reset	The actual position of the selected axis is set to zero (zero shift).
Import	The actual position of the selected axis is set to the entered value.
Correct	The actual position of the selected axis is shifted by the entered value.
S switchover	Enter the number of the workpiece coordinate system (S), complete with **<Enter>**, cancel with **<ESC>**. The entered coordinate system is activated.
back...	Back to the previous level.
Tool management...	Management of tools and the corresponding change positions. Up to 99 tools can be entered and saved. Tool ”0” serves as reference tool and should not be used otherwise. The tools can be assigned 99 tool magazine positions. Please note that each magazine position reduces the number of workpiece coordinate systems (max. 99). The number of tools and magazine positions managed in the MMI is determined in the ”DelphMMI.ini” (📖 327). 30 parameters are available for each tool, six of which have a fixed meaning: ● Tool length ● Tool radius ● Tool life ● Speed ● Magazine position ● Sister tools The data is made available in the parts program in the P field (P700 ... P729) when the tool is selected.

Tool data...		This key requests the current tool correction data from the NC computer and saves them in the file ”vom_nc.wtk” in the configurations directory. The received data is interpreted according to the specification in the tool management and displayed in the tool correction table. Each defined tool can be assigned a magazine position. It is not permissible to assign a magazine position more than once. A tool is regarded as defined if a tool number is entered in the ”Original tool” field. A sister tool is created by entering the number of the original tool in the ”Original tool” field of the sister tool. Data which has already been entered for the original tool can be adopted for the sister tool. When a field is quit, the entered values are checked. The field can only be quit if correct values have been entered. When the ”Original tool” and ”Sister tool” fields are changed, some of the fields are filled in automatically.
		 ETCN069
	Import	The data from the table is adopted in the NC computer; however, only temporarily, i.e. the information is lost when the control system is switched off.
	Import + save	The data from the table is transferred to the NC computer and saved on the PC in a file.
	back...	Back to the previous level.
Change positions...		This key requests the current magazine data from the NC computer and saves it in the file ”vom_nc.wsk” in the configuration directory. The received data is interpreted according to the specification in the tool management and displayed. The table shows the assignment between magazine positions and the coordinate systems. The change positions are also entered in the table. The positions for X, Y, Z and C can either be entered manually or determined via a teach function. If the machine offers the possibility to move the axes independently from softkeys, the positions can be approached and adopted for the selected magazine position by means of the **<Position import>** key. The data is entered in the corresponding columns.
	Position import	The current axis positions are read as tool change positions and entered in the table.
	Import	The data from the table is adopted in the NC computer; however, only temporarily, i.e. the information is lost when the control system is switched off.

 ETCN070	
Import + save	The data from the table is transferred to the NC computer and saved on the PC in a file.
back...	Back to the previous level.
Load tool record	Open a dialogue for selecting a file with tool data (WTK file). This file will be transferred to the control system.
Tool number	In the upper edge of the ”Status displays” dialogue there is a field for entering a tool number (Tx). After the key has been pressed, the field has a coloured background and the cursor flashes in this field. A tool number can be entered here. Changes are adopted with **<Enter>** and discarded with **<ESC>**.
back...	Back to the previous level.
Enter M function	In the upper edge of the ”Status displays” dialogue there is a field for entering an M function. After the key has been pressed, the field has a coloured background and the cursor flashes in this field. A function number can be entered here. Changes are adopted with **<Enter>** and discarded with **<ESC>**.The M function is transferred to the PLC for execution.
Entry	Enter single block (e.g. G, S, T and M functions). The execution is completed and triggered with **<Enter>**.

Vertical function keys

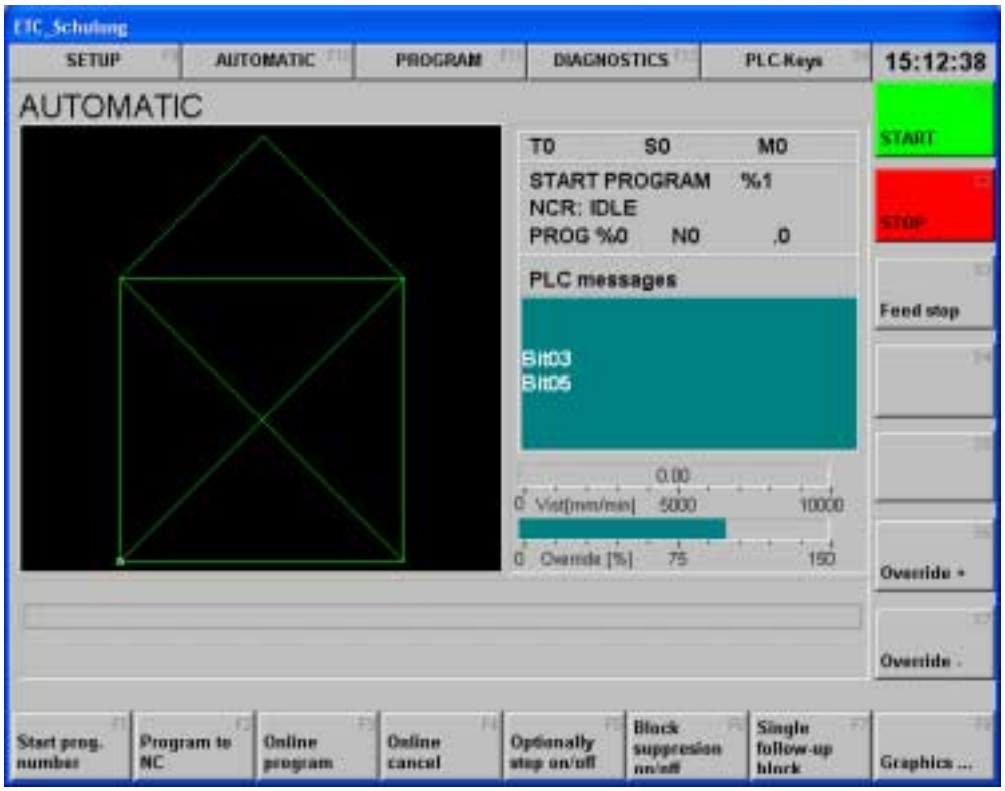
Start	Trigger action (e.g. automatic home position approach).
Stop	Quit all axis movements and a running program.
Travel+	Manual traverse key for positive axis direction of the selected axis.
Travel-	Manual traverse key for negative axis direction of the selected axis.
Axis	Select the axis to be traversed. Alternatively, the axis can be selected by means of the **<Cursor>** keys.
Override+	Increase the axis or path speed of the selected axis (in percent of MK_VMAX). Over 10%, the value is changed in steps of 10%, below 10% in steps of 1%.
Override-	Reduce the axis or path speed of the selected axis (in percent of MK_VMAX). Over 10%, the value is changed in steps of 10%, below 10% in steps of 1%.

Vertical function keys in the Tool management menu

When you switch to tool management, a new vertical softkey line is shown.

Fetch tool	The tool ”Tx” displayed in the status fields is fetched from the tool magazine and placed in the tool holder.
Change tool	The tool currently located in the tool holder is replaced by the tool ”Tx” displayed in the status fields. The functions of the tool change require a corresponding PLC program (DIN program).
Deposit tool	The tool ”Tx” displayed in the status fields is stored in the tool magazine.

7.5 ”Automatic” operating mode

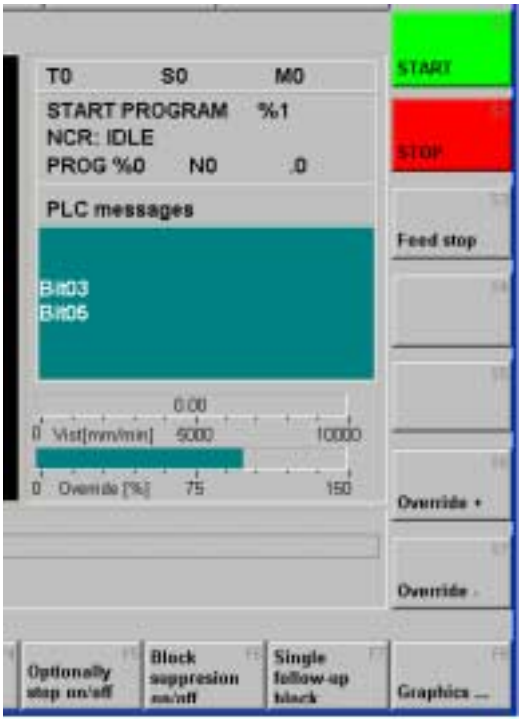


The ”Automatic” operating mode is always active when the NC computer regularly processes a program. It shows the most important information of the program flow.

Horizontal function keys

Start program number	Enter the program number of the program to be edited. Before, the program must be transferred via the ”Program to NC” function (if required, with all necessary subprograms). The program is started via the **<Start>** key.
Program to NC	Transfer program from the PC to the ETCxC. After a function has been called, a dialogue is displayed. Via the **<Cursor>** keys, select a program and start the transfer with **<Enter>**. **<ESC>** cancels the action. Note: In the ”DelphMMi.ini”, you can determine which programs are loaded automatically when the control system is initialised (📖 327).
Online program	Use this function for programs which are too big to be transferred to the NC as a whole. A dialogue opens (📖 306). Here you can selected the desired program. After **<Enter>** has been pressed, the program is transferred to the ETCxC block by block. With **<Start>**, the program is started without having to be transferred completely. The output line shows the selected program and the part (in percent) that has already been transferred. If the text ”WAIT” is displayed behind the percentage, the memory in the ETCxC can no longer accept any data at the moment. After the transferred block has been processed, the next block is automatically transferred until the program has been completely processed. When editing a program ”online”, please note that the program must not contain any jumps and branches. For subprograms or cycle calls, the program must be loaded in the ETCxC first.
Online cancel	Quit online mode. After a restart, processing must be started anew.
Optionally stop on/off	Stop the program with a ”M1” command. The program is continued with **<Start>**. M1 must be edited accordingly by the PLC.
Block suppression on/off	Do not execute all blocks marked with a ”/”.
Single follow-up block	Each traverse block must be triggered by the **<Start>** key. In the case of a switch-over during running operation, processing is stopped with the next possible block. After the ”Single block” function has been deactivated, the **<Start>** key must be pressed again.
Graphics...	
Graphics on/off	Show a graphical display of the current program instead of the position display. The graph is currently limited to a profile description of the axes X and Y. Zero shift, rotation, individual programming and similar are not supported.

Vertical function keys

Start	Starts the selected program or blockwise processing in single block operation.
Stop	Immediately stops program execution. All axes are stopped with the set deceleration ramps. After a restart, processing is started again.
Feed stop	If ”Feed stop” is activated, the axes are no longer traversed, until the function has been deactivated. The axes are stopped with the set deceleration ramps. The current traversing speed and the set override value are displayed in the status field. The function must be executed via the PLC. Either ”Feed stop” or ”Optionally stop” is active.  ETCN072
Override+	Increase the axis or path speed of the selected axis (in percent of MK_VMAX). Over 10%, the value is changed in steps of 10%, below 10% in steps of 1%.
Override-	Reduce the axis or path speed of the selected axis (in percent of MK_VMAX). Over 10%, the value is changed in steps of 10%, below 10% in steps of 1%.

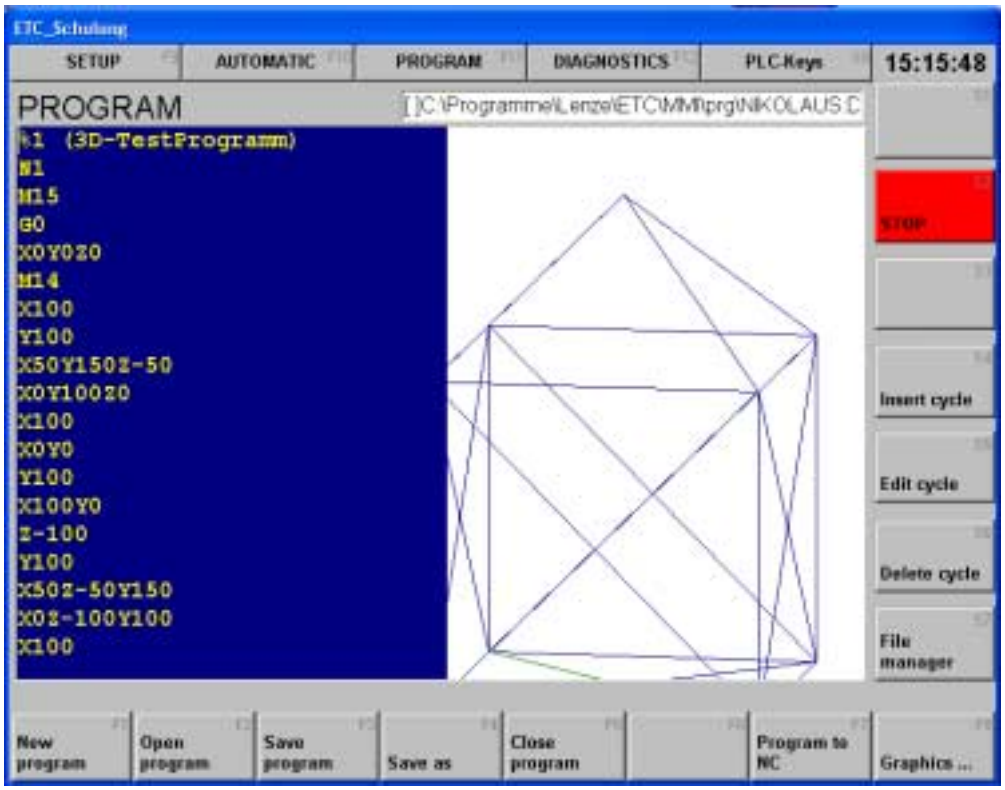
Horizontal function keys for graphics ”on”

Drawing - border	A frame is drawn around the DIN program and dimensioned in the input units.
Enlarge	Zoom into the profile
Reduce	Zoom out of the profile.
Zoom	The graphic is maximised. All other displays are hidden.

Vertical function keys for graphics ”on”

Display idle travel	Idle travels are displayed in a different colour (M15/M14). Note: The graphic detects idle travels by means of an upstream M15, work paths by means of M14.
Progress in colour	The colour of paths that have been travelled completely changes.
X plus	Shift representation in X plus direction.
X minus	Shift representation in X minus direction.
Y plus	Shift representation in Y plus direction.
Y minus	Shift representation in Y minus direction.

7.6 ”Programming” operating mode



ETCN073

In the ”Program” operating mode, you edit CNC programs or any text files.

The operating mode offers a file management system for copying, printing and deleting files (📖 308) and an ASCII editor (📖 305) for editing programs.

The name of the file being edited is displayed in the upper right. In the editor field (in the illustration on the left), the contents of the file are displayed and edited. For a CNC program, a graph can be optionally displayed (in the illustration on the right).

Up to eight files can be opened at the same time. The names are entered in a list. By means of the **<TAB>** key and the **<Cursor>** keys, the files are selected. By means of the **<Enter>** key, they are brought to the foreground.

The operation of the editor is described in a separate section.

Horizontal function keys

New program	Prepare the editor for entering a new file. If a file is already being edited, it remains active in the background. Via the file selection line in the upper window area, files from the background can be brought to the foreground.
Open program	Open existing (program) file. A dialogue opens (📖 306). Here you can select the desired file. After **<Enter>** has been pressed, the file is opened in the editor.
Save program	Save the displayed (program) file. If a new file has been created, a dialogue opens (📖 306). Here you can enter directory and file name.
Save as	Save displayed (program) file under a new name. A dialogue opens (📖 306). Here you can enter dialogue and file name. The contents of the original file remain unchanged.
Teach-IN...	This key is only visible if a corresponding bit is set in the PLC. Switch on teach mode. In the teach mode, the current position can be determined and transferred to the cursor position in the editor.
Teach-In ON/OFF	In this mode, the axes can be traversed via the PLC. By means of a PLC message, the current position is entered in the editor as traverse block with the axis letters of the axes configured for teach-in.
Teach-In Extended	In the extended teach-in mode it is possible to observe the program flow in single block operation in the editor. The current line position of the ETCxC is highlighted. For this purpose, the program currently running on the ETCxC must be loaded in the editor. The user can now control his program via the single block operation and teach in some blocks again for correction. In this mode, the blocks are overwritten if the cursor is not on an empty line. Otherwise, the blocks are added. **Caution:** When new blocks are added, the line number in the editor does no longer correspond to the line number in the running program of the ETCxC. This means that the wrong line position will be displayed during the following program flow.
Close program	Quit editing the program and delete editor contents. The original file is not changed.
Program to NC	Transfer program from the PC to the ETCxC. After a function has been called, a dialogue is displayed. Via the **<Cursor>** keys, select a program and start the transfer with **<Enter>**. **<ESC>** cancels the action. Note: In the ”DelphMMi.ini”, you can determine which programs are loaded automatically when the control system is initialised (📖 327).
Graphics...	Display (in 3 D mode) of the profile of a program which is in the editor. The graph is currently limited to a profile description of the axes X, Y and Z. Zero shift, rotation, individual programming and similar are not supported.
Reset image position	Show original position of the profile.
Display position	If a program is running, use ”Display position” to display and observe the current position on the profile.
Graphics on/off	Show and hide graphical display. The size of the window can be changed by means of the ”Split screen” function.
back...	Back to the previous level.

Vertical function keys

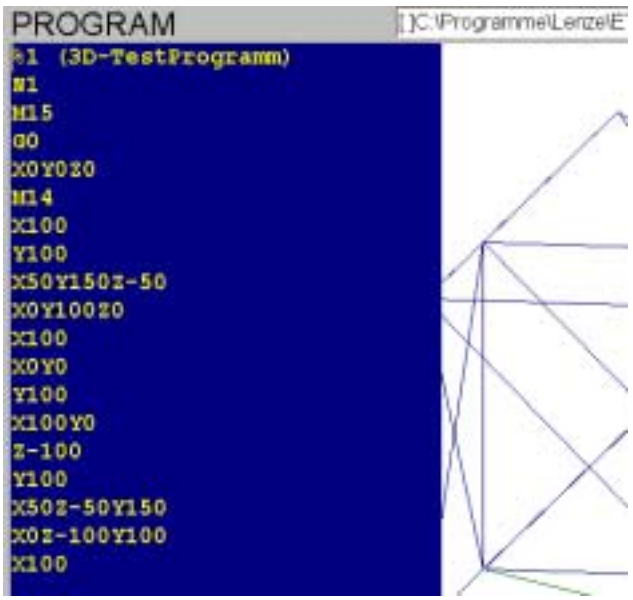
Stop	Quits a running program.
Insert cycle	Adds a prepared cycle. The cycle is selected by means of the **<Cursor>** keys and added to the line where the cursor is positioned by means of **<Enter>**. **<ESC>** cancels the selection. A cycle is a subprogram to which parameters are transferred when it is called. More detailed information on creating cycles can be found later in this chapter (📖 310.).
Edit cycle	Edit existing cycle. The cursor must be positioned on the line in which the cycle is called.
Delete cycle	Delete a cycle call at the current cursor position.
File manager	The exact description of the file manager including the meaning of the softkeys used can be found later in this chapter (📖 308.).

Vertical function keys for graphics ”on”

Rotate by X, Y, Z	Rotate the representation of the profile in X, Y and Z direction (via **<+>**and **<->**) keys.
Move by X, Y	Shift the representation of the profile in X and Y direction (via **<+>**and **<->**) keys.
Split screen	Change the size of the graphic window (via **<+>** and **<->**) keys.
Zoom	Change display size of the graphic (via **<+>** and **<->**) keys.

7.6.1 ASCI editor

The editor is primarily used for entering and changing CNC programs according to DIN 66025, but also for editing any ASCII files.



Note!

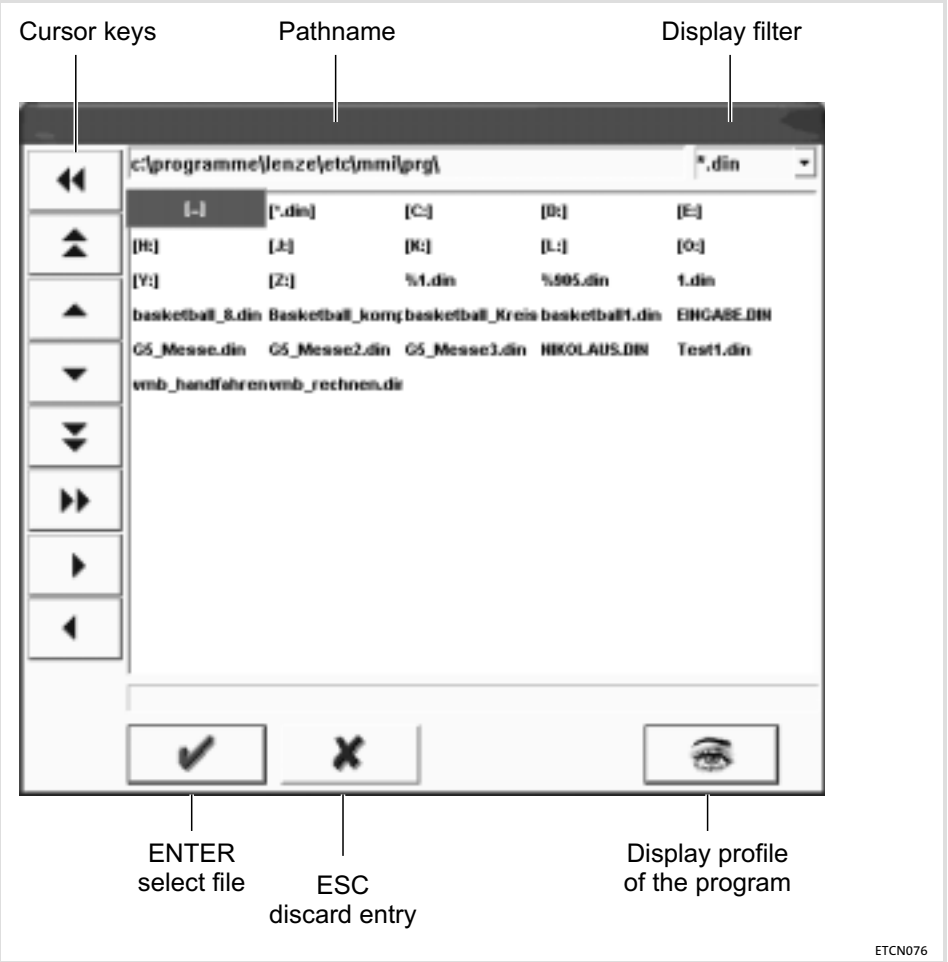
We recommend to operate the editor via an external keyboard. To facilitate program creation, the CNC blocks are automatically set in upper case letters. If you want to enter lower case letters (e.g. for axes u, v, w), press the **<Shift>** key during the entry. In comments, the editor automatically switches to the usual entry form.

Key functions

<Cursor left/right>	Move cursor a character to the left/right.
<Shift>+<Cursor left/right>	Highlight character to the left/right of the cursor.
<Pos1>	Set cursor to beginning of line.
<End>	Set cursor to end of line.
<Ctrl-Pos1>	Set cursor to beginning of file.
<Ctrl-End>	Set cursor to file end
<Page up>	Browse up page by page.
<Page down>	Browse down page by page.
<Ctrl>+<C>	Copy range to clipboard.
<Ctrl>+<X>	Move range to clipboard (cut).
<Ctrl>+<V>	Paste contents of clipboard at cursor position.
<Ctrl>+<Z>	Reset last change of a line.
<ENTER>	Add new line.
<Backspace>	Delete the character to the left of the cursor or marked text block.
<DEL	Delete the character to the right of the cursor or marked text block.

Dialogue box for file selection

This dialogue box is used, among other things, for the selection of a program for the editor and for the transfer to an NC computer.



General functions

- ► A dialogue box (button, input field, selection list etc.) can be selected with the **<TAB>** key.
- ► By means of the **<Cursor>** keys, you approach an element in the selected dialogue area (e.g. a certain file in the file selection area).
- ► With the **<Enter>** key, you execute an action. With **<ESC>**, you cancel the action.

Select file

1. Press the **<TAB>** key several times, until the display and input field of the directory name is selected.
2. If you want to change the file path, press **<Enter>** and enter a new path. Press **<Enter>** to confirm the entry or **<ESC>** to discard it.
3. Press the **<TAB>** key several times until the file selection field is selected.
4. Select the file by means of the **<Cursor>** keys. Press **<Enter>** to select the file.

Display profile of the program

1. Select the file whose profile you want to display (see above).
2. Afterwards, press the **<TAB>** key several times until the ”Kontur des Programms anzeigen” (”Display profile of the program”) button is selected.
3. Finally, press **<Enter>**.
   The profile of the selected program is displayed.
4. Press **<ESC>** to close the display.

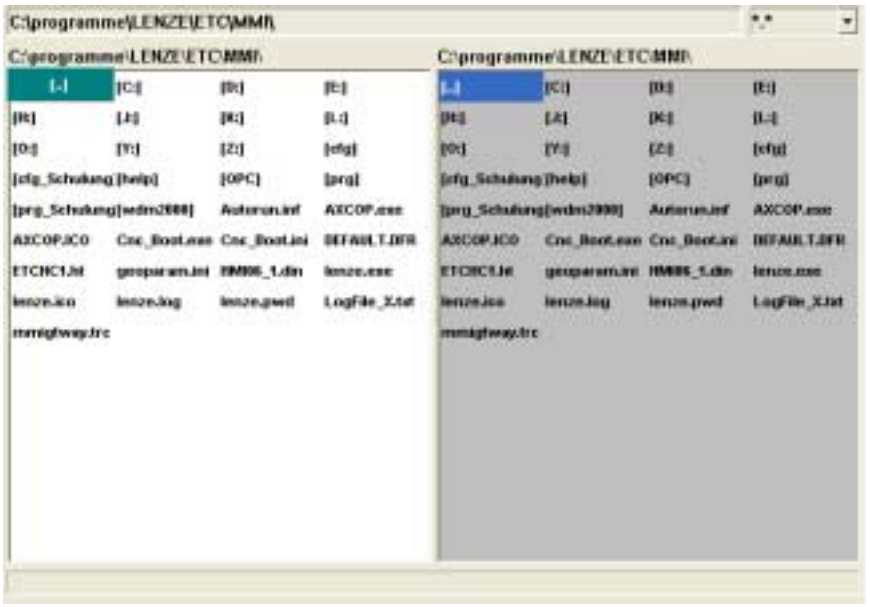
Set filter

By means of the display filter, the display of the files in the ”File selection field” can be extended or limited.

The display filter is shown in addition to the directories in square brackets [*.DIN]. When this entry is selected, all data of the current directory affected by this filter is transferred to the control system.

7.6.2 File manager

The file manager provides functions for managing files. When the file manager is called via the corresponding softkey of the ”Programming” operating mode, a new softkey line is shown, which provides all functions of the file manager.



ETCN077

In the file manager, different directories can be displayed independently from each other in the left and right file window. The active file window has a white background. In the ”Directory input field” the directory of the active file window can be determined. The file filter window makes it possible to filter the files in the specified directories for the display. The file filter is applied to both file windows.

In the upper area of the file manager, the available drives and directories are listed. An entry with two dots ([..]) is a reference to a higher-level directory. In the lower area, all files in the current directory are listed.

General functions

- ► A dialogue box (button, input field, selection list etc.) can be selected with the **<TAB>** key.
- ► By means of the **<Cursor>** keys, you approach an element in the selected dialogue area (e.g. a certain file in the file selection area).
- ► With the **<Enter>** key, you execute an action. With **<ESC>**, you cancel the action.

Horizontal function keys

Copy	Copy selected file(s).
Move	Move selected file(s) to a different directory.
Delete	Delete selected file(s).
Rename	Rename selected file(s).
Print	Print selected file(s).
Graphics	Display the graphic selected by the cursor. <ESC> closes the graphic.
Display	Display the file selected by the cursor (write-protected). <ESC> closes the graphic.
back...	Exit file manager.

Vertical function keys

Mark one	Mark the graphic selected by the cursor.
Mark all	Mark all files. If files are already selected, this function reverses the selection.

Hotkeys

c	Copy
d	Delete
m	Move
p	Print
r	Rename
v	View
+	Selection via filter
-	Deactivate filter selection
*****	Reverse selection
.	Dot, select/deselect current file

7.6.3 Cycle programming

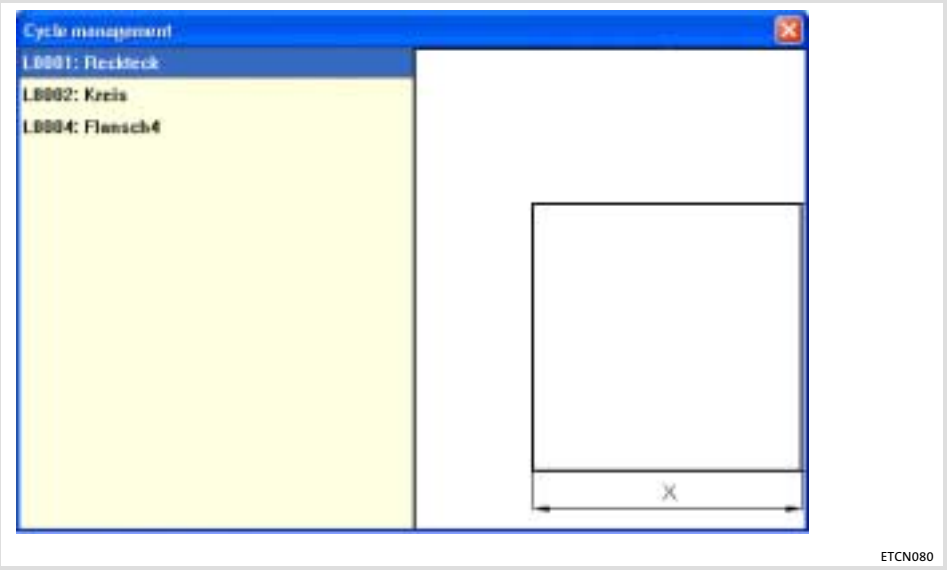
A cycle is a subprogram to which parameters are transferred when it is called. During cycle programming, the parameters are requested by the user and made available to the DIN program in the P fields. Existing cycles are defined in the configuration file. More detailed information on creating cycles can be found later in this chapter (📖 325).

Insert cycles

When the ”Insert cycle” function in the ”Programming” operating mode is selected, a window with a list of available cycles opens. From this list, the cycle to be added to the displayed DIN program can be selected.

After **<Enter>** has been pressed, the parameters belonging to this cycle are displayed. **<Enter>** takes you from input field to input field. **<Enter>** in the last field enters the cycle as subprogram call in the editor field.

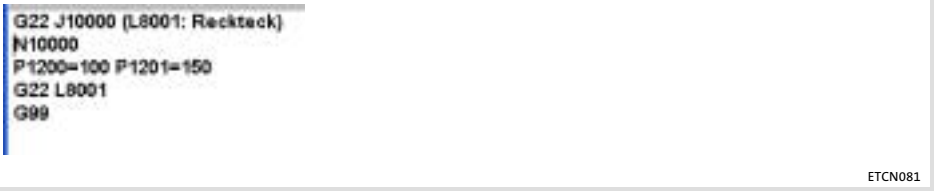
The actual program call and the transfer of the parameters are written to the end of the program starting with block number 10000.



Example

```
%1
G22 J10000 (L3501)
M30
(*** The cycles will follow from
here ***)
N10000
P1200=100.0000    P1201=100.0000    P1202=20.0000
```

Representation of the cycle in the DIN program

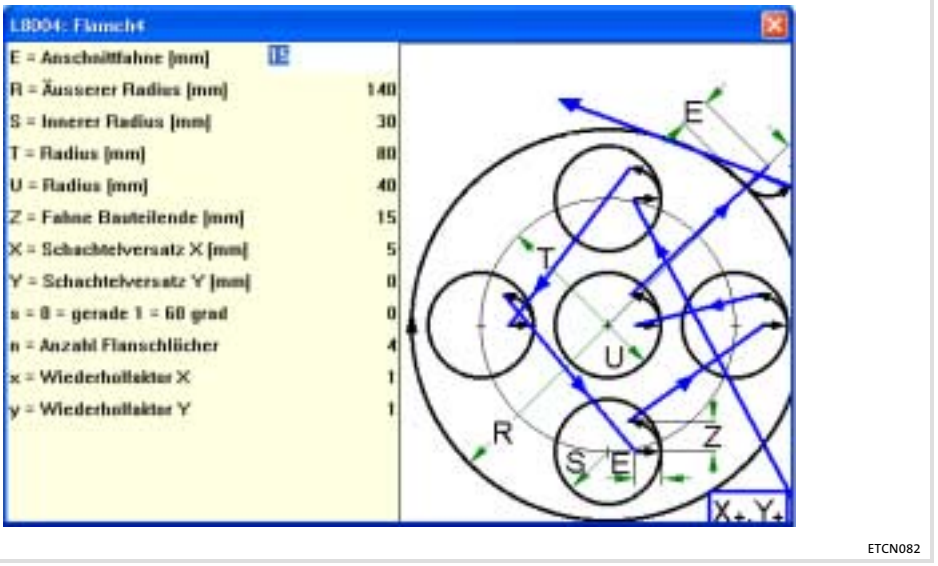


P1203=20.0000
P1206=0.0000
G22 L3501
G99

P1204=20.0000
P1200=90.0000

P1205=1.0000

Edit cycles



After a function or a cycle has been selected, the respective entries are made in the DIN file.

Change cycle

If the cursor is positioned on a line starting with ”G22 Jxxxxx”, the cycle can be edited and the data changed as long as all information is available.

Delete cycle

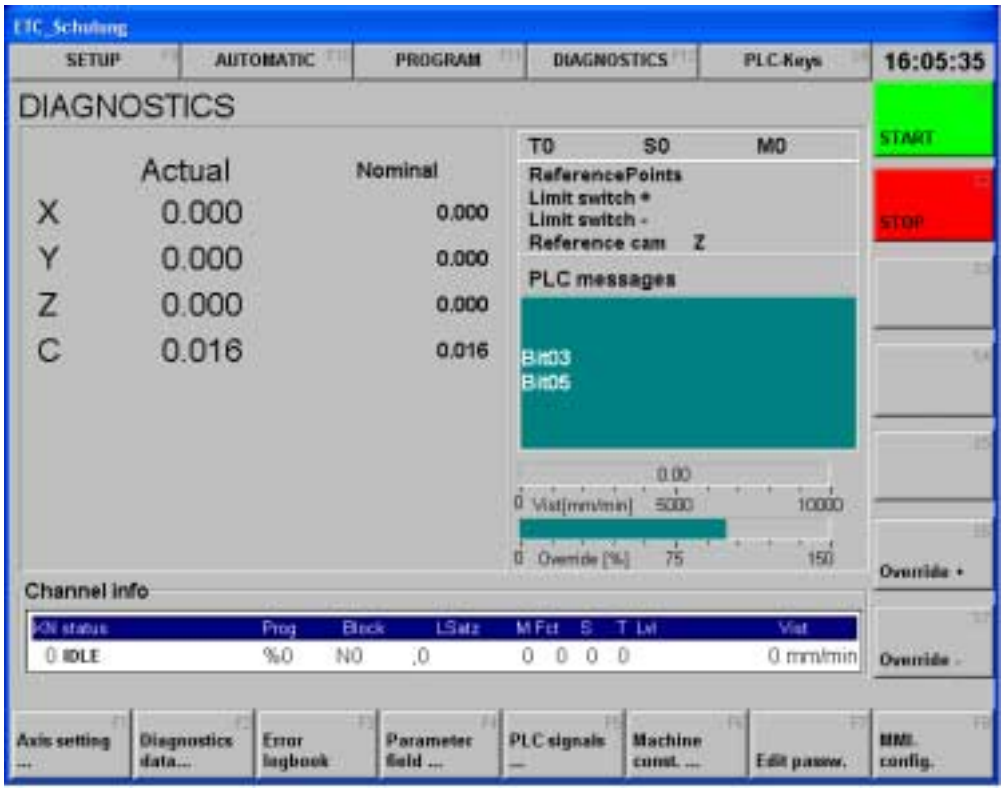
If the cursor is positioned on a line ”G22 Jxxxxx” in the editor, the cycle (call and parameter part) can be removed from the DIN file.

7.7 ”Diagnostics” operating mode

The ”Diagnostics” operating mode is primarily intended for service and commissioning technicians.

It offers functions for the support of axis setting, for the control of inputs/outputs and parameters, for editing machine constants and for the control of internal statuses.

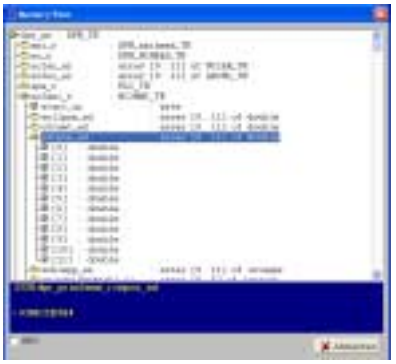
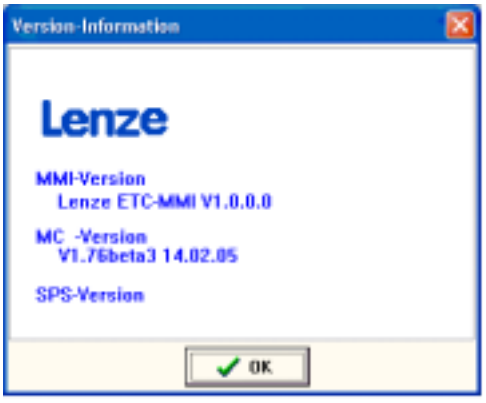
In addition, the error logbook, which logs all errors, can be accessed.



ETCN083

Horizontal function keys

Axis setting...	Activate displays for support during axis setting. The handling corresponds to the SETUP operating mode.
Modal travel	The axis selected with the **<Axis>** key (or <Cursor> keys) is traversed as long as the **<Travel +>**/**<Travel ->** key is pressed. The speed at which the axes are traversed is determined by MK_MODVMAX. It can be influenced via the **<Override +>**/**<Override ->** key.
Drag/Mod/Nom p.	Switch-over of the display between: ● Tolerance margin in input unit [mm] ● Set position ● Modal position
Input	Enter single block (e.g. G, S, T and M functions). The execution is completed and triggered with **<Enter>**.
back...	Back to the previous level.

Diagnostics data...	
NC timing information	Display different internal control-specific data. The function is intended for internal use only or for trained service personnel. The display can be cleared with the **<ESC>** key.
Trace	Open dialogue for activating events that are to be logged. From the list of available traces, individual traces can be selected with a double-click and activated with **<Start>**. You will then be prompted to enter a file name (e.g. \log\ETC_MMI.trc). The events will be written to this file. The activations will be reset when the MMI is quit.
Axis information	Show parameters of the configured axis.
Dump	The file ”IBprot.txt” is created in the ”\log” directory and displayed. It contains information on the system (inter alia DelphMMI.ini, current machine constants, current tool data, contents of the P fields). This way, relevant data can be collected if problems occur.
DPR	Display the structure of the complete DPRs with the current values.  ETCN084
Version information	Version information on the ETC-MMI, firmware etc.  ETCN085
back...	Back to the previous level.

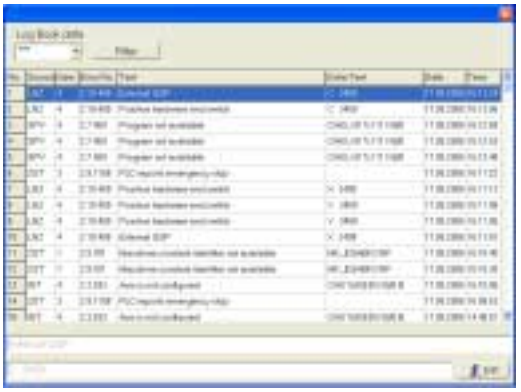
Error logbook

If errors occur in the communication of PC, PLC, NC and/or during programming, they are logged together with date and time in a file (errorlog.txt). By means of the ”Error logbook” function, the error messages can be displayed. Use the cursor keys to browse through the individual fields. With **<ESC>** or **<EXIT>** the display can be quit. Errors older than 30 days are removed from the logbook. The individual columns have the following meanings:

- **No:** sequential number
- **Source:** Indicates the module that reported the error.
- **Sev:** Indicates the severity of the error (e.g. 1 = information; 4 = fatal error causing an abort).
- **Error no:** Clear identification of the error
- **Text:** Error text from the error text files ”*_Fehl.DB”
- **Additional text:** Additional information of the module, which gives an indication of the cause of the message.

The texts of the currently selected error are displayed again in the lower part. If e.g. an additional text is not completely visible in the table, it can be read here.
By means of the source filter, a module is selected whose errors are to be displayed. Select *** to display the messages of all modules.

Representation of an error log without filter:

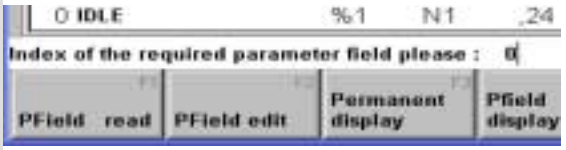
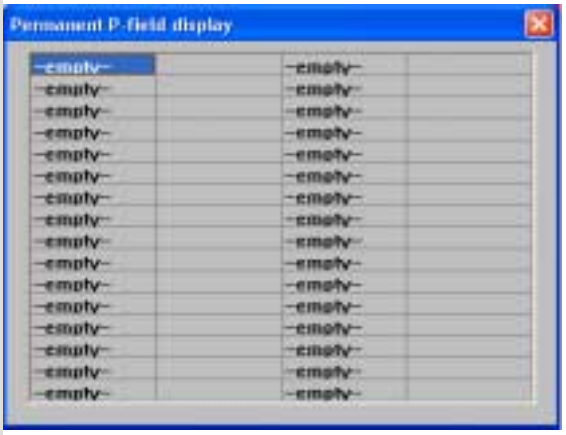


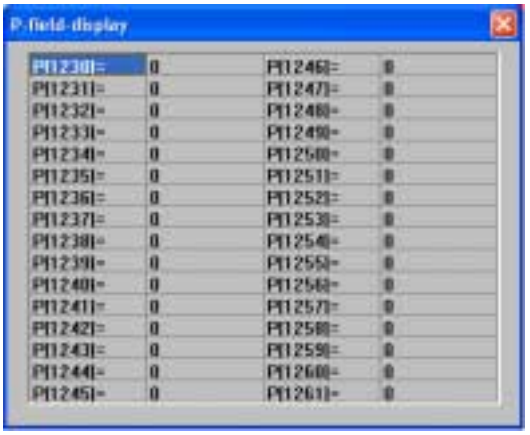
ETCN086

Representation of an error log with filter:

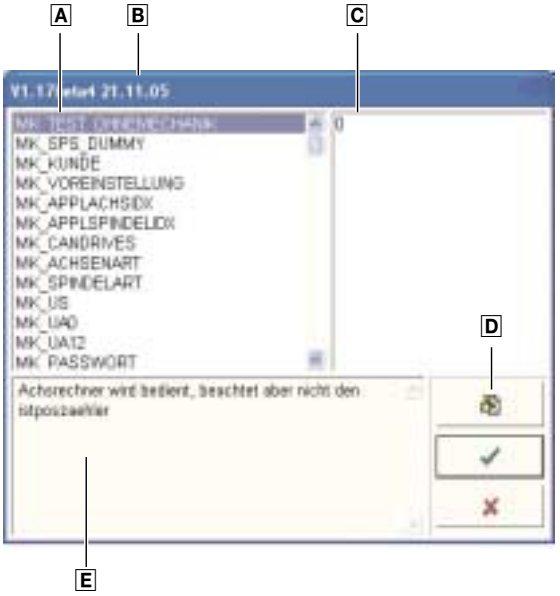
ETCN087

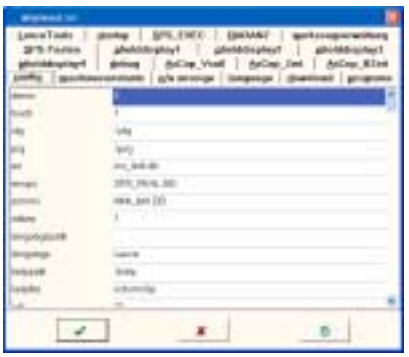
When filters are used, additional options for filtering the messages are offered; for example
for a period of time or the number of messages to be displayed. With **<Import>**, the filter settings are adopted. With **<Exit>**, the extended filter is closed.

Parameter field (P field)	The following applies for all entries: The **<Enter>** key is used to confirm the entry, the **<ESC>** cancels the entry.
P field read	Display the value of a parameter field. The P field number is entered in the status line. The contents are output in the display line. If this field is selected, the number that was entered last is automatically offered. A different number can be entered immediately without having to delete the old display.  ETCN088
P field edit	Change the value of the desired parameter field. Both entries are made in the input line. For entering the digits, the same applies as to ”P field read).
Permanent display	Opens a separate window with a list of 32 parameters. They are displayed in their current version until the dialogue window is closed again with **<ESC>**. Entered P fields are permanently displayed in this window. The parameters are selected by selecting a position by means of the cursor keys and opening an input window by means of the **<Space bar>**. There, the desired P field number is entered. When the window is called, the P fields specified in the configuration file are automatically displayed. These numbers can be overwritten at any time.  ETCN089

	P field display	Opens a separate window with a list of 32 successive parameters. The start parameter can be entered by pressing the **<Space bar>** or **<Enter>** key. When the last P field is selected, the P field index is displayed. Press **<Enter>** to display this index in the first field. This function can be used to browse 31 fields up. If **<Enter>** is pressed on a field with value of the P field, this value can be changed via an entry block. The dialogue can be closed with **<ESC>**.  ETCN090
	back...	Back to the previous level.
PLC signals		This function offers different options of checking the communication between MMI and PLC and monitoring the statuses of the inputs and outputs of the PLC. 📖 240.
	MMI <> PLC	Opens two windows showing the data exchange between the MMI and the PLC via the data block DB2. The upper window shows data sent by the MMI program to the PLC. The lower window displays data sent by the PLC to the HMI. Use the **<TAB>** key to switch between the windows and use the **<Cursor>** keys to scroll down the display. ETCN091

Enable PLC	The Enable PLC function is used for checking the most important interface signals between PLC and NC. ETCN092
Remote On/Off binary	Displays the input and output statuses of the external I/O modules connected via CAN bus in binary form (0 = input/output not set). This function is only significant if the system is equipped with the corresponding hardware.
Cards On/Off ...	
Local dig. I/O	In the language file, a text can be saved for each input and output (📖 321). If an input is set, it is displayed in a different colour.
local anal. I/O	
Remote I/O	Shows a list of the configured CAN bus modules. A module can be selected by means of the **<Cursor>** keys. Press **<Enter>** to display the inputs and outputs of this module.
back...	Back to the previous level.
back...	Back to the previous level.

Machine constants	Edit machine constants (📖 190).  ETCN093 A: Available machine constants B: Date and version of the NC C: Setting of the current machine constant D: Change value E: Description of the machine constant from the ”MK.hlp”. Individual parameters can be changed in the machine constant catalogue.
edit MC	When ”edit MC” is selected, the current machine constants are first loaded from the NC computer and then displayed. They can now be changed. Use the **<TAb>** key to select the individual elements in the dialogue box. If you are in the field in which the machine constants are listed, you can make the setting(s) for the selected constant by pressing the **<Space bar>**. Entries must be confirmed with **<Enter>**. Use the **<ESC>** key to close the dialogue for changing the machine constants without adopting the changes. With **<Enter>**, you can quit editing. Afterwards, the changes are sent to the NC and updated in the configured MC file. Thus, the changed machine constants will be loaded again during the next start.
load MC	Show available machine constant files. To transfer a file to the ETCxC, select it with the **<Cursor>** keys and press **<Enter>**. Note: During the next start of the control system, the MC file entered in the INI file will be loaded.

EDS	Load description file of a manufacturer of a CAN module. All defined code positions can be read out. The code positions released for writing can be overwritten with new values. For details, refer to the module description of the manufacturer of the CAN modules. If the softkey is pressed, a selection of the CAN bus (1:I/O bus, 2:drive) is displayed first. Afterwards, the node number of the desired module must be specified. In the following dialogue, the description file (*.eds) of the module type is selected. The selected module can now be accessed. ETCN094
back...	Back to the previous level.
Edit password	After the softkey has been pressed, a dialogue is displayed. The password for changing passwords must be entered here (📖 292.) Afterwards, a selection list is displayed where the password to be changed must be selected. You will then be prompted twice to enter the new password in order to avoid possible errors. New passwords become active immediately after the change. If you do not want a password for certain operating modes, enter an empty string as active password.
MMI configuration	Display and change configuration file ”DelphMMI.ini” (📖 327). Every tab represents a section of the Ini file. Use the button in the right bottom to add a new entry below a section. It can, for example, be used to enter cycles in the ”programs” section. These cycles will be loaded on startup.  ETCN095

Vertical function keys

Start	Starts the selected program or blockwise processing in single block operation.
Stop	Immediately stops program execution. All axes are stopped with the set deceleration ramps. After a restart, processing is started again.
Travel+	Manual traverse key for positive axis direction of the selected axis.
Travel-	Manual traverse key for negative axis direction of the selected axis.
Axis	Select the axis to be traversed. Alternatively, the axis can be selected by means of the **<Cursor>** keys.
Override+	Increase the axis or path speed of the selected axis (in percent of MK_VMAX). Over 10%, the value is changed in steps of 10%, below 10% in steps of 1%.
Override-	Reduce the axis or path speed of the selected axis (in percent of MK_VMAX). Over 10%, the value is changed in steps of 10%, below 10% in steps of 1%.

7.8 Appendix

7.8.1 Language file (SPRACHE.TXT)

The language file (e.g. SPRACHE.TXT) contains all display and message texts, with the exception of the error messages, which are saved in separate files. The file is created in the ANSI character set typical of Windows and can be edited with any editor (e.g. the editor in the PROGRAMMING operating mode of the MMI software).

The beginning of a text line is always marked by a number. The number serves as a selection criterion for the MMI software. If there is no number at the beginning of a line, the text is added to the previous text with word wrap. Some of the texts contain additional information, which are not directly displayed.

- In addition to the text, the softkey labels also contain the information on the colour of the softkeys (@C), an optional bitmap, its alignment and possible animation (@B), the text flow, the font to be used, its size and colour (@F). For example:

  00000000,”Menu@C12632256@BFLDRMANY,4,4,2,1,1,0,0@FArial,8,1,0,3,6”

- If a tilde (”~”) is added to a text, a statement ”%Axx~” in the corresponding text is replaced by the code letter of the axis marked with ”xx”. The code letter results from the current machine configuration. The index of the axis is determined by the entry ”xx”. If the machine configuration is ”XZC”, the text ”This is the %A2 axis~” is displayed as ”This is the C axis”.

The numbers -1 ... 00001323 are used for labelling the function keys (softkeys). Numbers below 3000 may only be used for function keys, not for other displays or similar.

Examples from the language file

```
-0000001,”@C12632256@FArial,8,1,0,3,6” <- empty function key
00000000,”Menu@C12632256@BFLDRMANY,4,4,2,1,1,0,0@FArial,8,1,0,3,6”
00000001,”Setup@C12632256@FArial,8,1,0,0,15359”
   .
   .
   .
00001312,”back...@C12632256@FArial,8,1,0,3,6”
00001323,”Delete cycle@C12632256@FArial,8,1,0,3,6”
.
.
00002800,” 1st PLC key@C12632256@FArial,8,1,0,3,6” <- labelling for optional PLC keys
00002801,” 2nd PLC key@C12632256@FArial,8,1,0,3,6” (see section [SPS-Tasten] in
DELPHMMI.INI)
.
.
00002926,”127th PLC key@C12632256@FArial,8,1,0,3,6”
00002927,”128th PLC key@C12632256@FArial,8,1,0,3,6”
```

From no. 3000, general texts follow, both for additional keys and notes, entries etc.

```
00003000,"Your password please :"
00003001,"Setup password"
.
.
.
01000007,"mc"
01000008,"File has been changed."
"Save change?"
01000009,"Cycle files cannot be loaded as job."
01000010,"The value of the entry is impermissible."
"For this parameter, only a value between %f and %f is permitted."
01000011,"Input fields"
```

The texts no. 01010000 ... 01010064 are used for the "PLC signals -> "Local On/Off text" function in the DIAGNOSTICS operating mode. This way, the inputs and outputs of the control system can be assigned clear names and designations (E 0.0 ... E 3.7 are the inputs).

```
01010000,"Basic IOs"
01010001," E 0.0 here e is 0"
01010002," E 0.1"
.
01010031," E 3.6"
01010032," E 3.7"
01010033," A 0.0"
01010034," A 0.1"
.
.
01010063," A 3.6"
01010064," A 3.7"
```

The texts no. 01010100 ... 01010164 are used for the "PLC signals -> "Remote On/Off text") function in the DIAGNOSTICS operating mode. This way, the external CAN bus I/O modules can be assigned clear names and designations (E 0.0 ... E 3.7 are the inputs).

```
01010100,"Remote IOs (CAN bus)"
01010101," E 0.0"
01010102," E 0.1"
.
.
01010131," E 3.6"
01010132," E 3.7"
01010133," A 0.0"
01010134," A 0.1"
.
.
01010163," A 3.6"
01010164," A 3.7"
```

From number 01020000 (to s 01020255) the PLC messages follow. They are displayed in the MMI software in the operating modes SETUP, AUTOMATIC and DIAGNOSTICS in the "SPS-Meldungen" ("PLC messages") window. The messages must be specified consecutively, i.e. there must not be any gaps in the numbering.

The PLC can output the messages by setting or deleting the bits DB 80.0 to DB 95.15 on the data block DB2 of the dual port Ram interface. In the MMI software, a message is output if the corresponding bit is set. If the text is preceded by a "!"t, the PLC message window changes its background colour, when this message is output.

It is useful to save these texts in a separate language file (see chapter configuration file: language). The PLC texts can be maintained independently from the MMI texts.

```
01020000,""
01020001,"Bit01"
01020002,"Bit02"
01020003,"!Bit03"
.
.
.
01020027,"Bit27"
01020028,"Bit28"
01020029,"Bit29"
```

From no. 1020300 general texts, error messages etc. follow.

```
01020300,""
02000000,"Problem during machine constant transfer"
02000001,"Too many files open at the same time"
   .
   .
   .
20000005,"Step travel   (increment axis +/-)"
20000006,"Modal travel    (axis +/-)"
```

Card-specific I/O display

In the path diagnostics\PLC signals, the "F6" softkey can be used to select the card-specific display of the digital and analogue inputs and outputs. The input/output modules are divided into 3 categories. Use the following softkeys to display the connected modules of the selected type in a list box.

Softkey "F1": The local digital input/output modules are displayed.

Softkey "F2": The local analogue input/output modules are displayed.

Softkey "F3": The remote input/output modules which can be addressed via the CAN bus interface are displayed.

A card is selected by "ENTER" or with a "double-click". The I/O display is built up according to the card configuration. The display contains lines and columns.

Digital inputs: The ON state is marked by a change in colour.

Digital output: A digital output can be selected via the cursor keys. By means of the ENTER key, the output is either switched on or off.

Analogue inputs/outputs: These displays are shown in two colours. One column shows the description of the input or output. In the other column next to it, the assigned value is displayed. The value is displayed as non-standardised and unsigned 16 bit value.

You can exit the mask by pressing the ESC key or by selecting the above-mentioned softkeys.

Text assignment of the inputs/outputs in the language file

The text assignment of the individual inputs and outputs is realised via text numbers. The text numbers are structured according to the following key. Each character corresponds to a digit.

Key	TKKKXNNN (8-digit number)
T	Type 0: local digital I/O module (EC-IO) 1: local analogue I/O module (EC-ADA, EC-ADC) 2: CAN I/O module (SLIO, CANOpen)
KKK	Node number 001 ... 127 for CAN bus modules 001 ... 008 for local modules
X	Type of the I/O signals 0: unused 1: digital input 2: digital output 3: analogue input 4: analogue output
NNN	Number of the I/O signals 000 ... 063 for CAN bus digital I/O signals 000 ... 007 for CAN bus analogue I/O signals 000 ... 023 for local inputs 000 ... 015 for local outputs

A basic number of 30000000 is added to this number. This basic number can be changed via a parameter in the configuration file. This way, the texts in the language file can be moved.

Example

Text in the language file: 30011000," E 0.0 here e is 0"

30000000	T = 0	The I/O signal texts start at this number.
1	KKK = 1	Card number 1. Local card
1	X = 1	Digital input
000	NNN = 000	Number of the digital input

Softkey assignment

The following text numbers are assigned to the softkeys:

```
00000037,"Cards On/Off ...@C12632256@FArial,8,1,0,3,6"
00001384,"Local dig. I/O@C12632256@FArial,8,1,0,3,6"
00001385,"Local anal. I/O@C12632256@FArial,8,1,0,3,6"
00001386,"Remote I/O@C12632256@FArial,8,1,0,3,6"
```

Heading assignment

The following text numbers are used for headings and labels:

```
00011200,"IO display :"
00011201,"digital In"
00011202,"digital Out"
00011203,"analog In"
00011204,"analog Out"
00011205,"EC - IO"
00011206,"EC - ADA"
00011207,"CAN"
00011208,"Value"
```

Settings in the configuration file ”delphmmi.ini”

In the configuration file ”delphmmi.ini”, the following parameters can be set in the ”e/a anzeige” (”I/O display”) section:

textbasisnr: Basic number where the I/O signal texts start.

separate_karten_texte: Can be used to determine whether separate texts are to be entered for each I/O module.

1: Separate texts for each module
0: The texts of module 1 are used for all modules

Example

[e/a anzeige]

textbasisnr=30000000

seperate_karten_texte=1

7.8.2 Cycle programming

For each cycle, an entry with the specification of the cycle number in the form of a section name (Lxxxx, where 8000 ≥ xxxx ≥ 9999) and a clear text name must be created. This name is displayed in the ”Programming” operating mode when the cycle is selected.

In the Lxxxx sections, the input parameters to be offered are determined. In addition, a bmp file can be specified. It will be displayed when the cycle is selected.

An input parameter is defined by a P field number, in which the input value is transferred to the cycle.

- ► Default value
- ► Minimum value (optional)
- ► Maximum value (optional)
- ► Identification for mandatory input (optional)
- ► Display text

enty(n) = [pfeldnr.] = x.xx; [Min]; [Max]; [Flag]; [Text]

In the following, some examples of cycles and their parameters are shown:

Section	Term	Description
[Zyklen]		
	entry1 = L8000; grinding	entry1: is required for identification. L8000: section name under which the cycle and the cycle number are described. grinding: text for display and selection
	entry2 = L8001; sawing	
	entry3 = L8002; sewing	
	entry4 = L8003; free	
	entry5 = L8004; free	

Section	Term	Description
[L8000]		
	entry1 = P1200 = 1.23; top	entry1: for identification P1200: definition of the corresponding P field 1.23: value which is set in the P field top: text for display and selection
	entry2 = P1201= 2.34; bottom	
	entry3 = P1202= 3.45; middle	
	entry4 = c:\stdmmi\cnc20.bmp	Image file that can be displayed
[L8001]		
	entry1 = P1200= 1.23; Min; Max; Flag; top	P1200=: definition of the corresponding P field 1.23;: value which is set in the P field Min;:minimum value that may be entered Max;: maximum value that may be entered Flag = 1: The value must be entered top: text for display and selection
	entry2 = P1201= 2.34;0;3.14;; bottom	In this example, the software checks the input for min/max. If there is no input, the field is not entered in the DIN file.
	entry3 = P1202= 3.45; middle	
	entry4 =	
[L8002]		
	Entry0 = c:\ProgramsLenze\ETC\mmi\Lenze.bmp, 25	Name of an image file that can be displayed. The 25 indicates that not half the screen width, but only 25% of it is available for the graphic. The value can be entered in the range 0 ... 100%.

Language independence

To ensure language independence, the display texts can also be taken from a file other than delphmmi.ini. The source file is the file specified in the INI file (e.g. delphmmi.ini) in the [config] section in the [language] entry (without file extension ".txt").

Example

Section	Term	Description
[Zyklen]		
	entry1 = L8000; #4100	The text with the no. 4100 from the file sprache.txt is displayed here.
[L8000]		
	entry1 = P1200= 1.23; Min; Max; Flag; #4151	The text with the no. 4151 from the file sprache.txt is displayed here.

7.8.3 Configuration file (DELPHMMI.INI)

The configuration file (by default delphmmi.ini) contains settings which are required for the operation of the IPC and the NC computer and should only be changed by trained personnel.

Definitions

A section refers to the expressions that are enclosed in ”[ ]” (example [config]). A term means the expressions following a section (example: cfg=c:\Programs\Lenze\ETC\mmi\cfg).

Please note that it is also possible to specify all paths within the configuration file relatively to the path of the executable program. This means that if the executable program is located under ”c:\Programs\Lenze\ETC\mmi”, the entry ”.\cfg” refers to the same subdirectory ”c:\Programs\Lenze\ETC\mmi\cfg”.

Contents

Section	Term	Description
[config]		
	demo=1	1 -> Operation without control system (demo installation) 0 -> Operation with control system
	mmiTrace=0	Log trace logs of the operating software. Is used by the manufacturer for troubleshooting.
	axformat=%.3f	Formatting instruction for the representation of the axis display in order to show more or less positions after the decimal point.
	nc=ETC_Kunde	Identification indicating which control system is to be addressed. The basic setting takes place automatically during installation (see ETC-MMI gateway). The name of the control system is set in the MMIGTWAY.EXE.
	touch=1	Deactivate touch screen display
	cfg=.\cfg	Name of the directory where all files for configuration are located. In the directory specified here, the error file with the name **errorlog.txt** is created.
	prg=.\prg	Name of the directory where DIN programs are located.
	err=ncr_fehl.db	This ASCII file (directory as specified under Term=cfg) contains error texts affecting the NC system.
	errsps=sps_fehl.db	This ASCII file (directory as specified under Term=cfg) contains error texts produced by the PLC.
	errmmi=mmi_fehl.db	This ASCII file (directory as specified under Term=cfg) contains error texts affecting the IPC.
	initbta=2	The value specified here determines the operating mode of the IPC, which is shown after a successful program start. 1: Setup 2: Automatic 3: Programming 4: Diagnostics

Section	Term	Description
	LanguagePath	Path in which the program looks for the language files.
	language=language	Here you can specify the name of an ASCII file (name extension ".txt" is presupposed). This file contains general display texts, additional error messages, the function key assignment etc. required for the operation of the MMI software (see also "language file"). If no entry is made here, a list of files can be entered in a language section. The specified text numbers must not be assigned twice. Note: All ASCII files can be edited with any editor (also with the ETC MMI software). You can thus adapt all texts in the files to your requirements (e.g. different language).
	left=0	Left window position if doscal <> 0
	top=0	Top window position if doscal <> 0
	doscal=0	Doscal = 0 means that the software is presented optically as it was programmed. doscal <> 0 can be used to change the window size of the MMI software. Doscal = 1 means that the complete screen is made available for the application. The left upper edge is defined by the terms left and top. Doscal = 2 to doscal = 1000 means that the window size can be changed from 0.2% to 100% of the complete screen size. Caution: The window can be minimised such that it can no longer be seen.
	watchdog=0	Watchdog <> 0 set offers the possibility to cyclically check whether the IPC communicates with the PLC, i.e. whether the connection still exists. For the operability, the running PLC program must support the watchdog function. As the watchdog function is not supported by the PLC sample program spsdummy.loc, it is deactivated by the statement = 0.
	QuitMode=0	If the value is 1, Windows is shut down after the program has been quit. If the value is zero, Windows is not shut down.
	DisplayFontSize=6	
	HelpPath	Specifies the path to the help files.

Section	Term	Description
	helpfile=c:\Programs\Lenze\ETC\mmi\ncform31.hlp	This term offers the possibility that the MMI software shows online helps. If the file name of a Windows help file is specified, the online help is displayed in Windows format. If only a directory (e.g. c:\Programs\Lenze\ETC\mmi\help) is specified, the standard software shows help texts in the form of ASCII files. The ASCII files must be located in the specified directory. In the MMI software, the function-related help is opened by simultaneously pressing the ”Control” key and a function key. The name convention of the file that is displayed is <number-of-the-function>.txt.
	NewTast=0	To activate the key of the IPC with the second function without keyboard driver, NewTast must be set to 1. By default, the keyboard driver is installed and NewTast = 0.
	Scanzposfile= testaufz.txt	Default name of the ASCII file in which the position values are saved in ASCII format.
	Zposkorrfile= testaufz.kor	3D correction file is transferred to the control system.
	saveanddownload=0	If SaveAndDownload is set to 1, a program is transferred immediately to the control system when it is saved during programming.
	startcycle=12345	This term is used during programming of cycles (see also ”cycle programming”).
[machineconstants]		
	mk=default.mk	This file contains machine constants that are transferred to the NC computer. The transfer only takes place if the HMI software is started and no operating system is running on the NC computer. If required, the NC computer must be reset to transfer the machine constants.
	mkhlp=mk.hlp	This file contains help texts that are displayed when machine constants are changed by means of the MMI (Diagnostics operating mode – Machine const. function) -> edit MC).
[download]		
	entry1=ETCxC; .\cfg\ETCxC.rsc;	Operating system for the NC computer: The operating system is only transferred after a reset of the NC computer.
	entry2=ncr; .\cfg\spsdummy.prg;	PLC program: The PLC program is only transferred after a reset of the NC computer. Note: After each installation, a manual entry is required here. You must specify the PLC file that you use here.
[programs]		
	entry0=d:\demo\std\cfg\9000.zyk	DIN program: If you want to load multiple DIN programs, multiple terms (entry1=..., entry2=... etc.) must be specified. The program is only transferred after a reset of the NC computer.

Section	Term	Description
[startup]		In the [startup] section, additional procedures during the startup of the HMI software are determined.
	dinLoad=1	This Boolean value defines (=1) whether after a complete (!) download of the NCR the automatic DIN program that was used last is loaded back into the control system and prepared for the program start.
	dinFile=c:\prog\beispiel.din	The name of the automatic DIN program that was used last. This entry is updated by the HMI.
	StartPrg=	Enter the number of the program (1 ... 7999) that is always to be started. If no number or zero is entered, the program that was loaded last is started.
	AxeKorr=	During program start, a leadscrew pitch correction file is transferred.
	logo= c:\bilder\logo.bmp	Specifies a bitmap that is to be displayed. If no file is specified, the default bitmap of the application is displayed.
	wait=1	The integer value (in seconds) determines how long the application is to wait while the logo is being displayed (>0), until it continues to start up the system. The value is limited to the entry under the term ”time”.
	time=5	Specifies the period of time (in seconds) that the logo is displayed. If the value is 0, the display of the logo is suppressed.
	Top=10	Defines the Y coordinate of the left upper edge of the image in the display.
	left=10	Defines the X coordinate of the left upper edge of the image in the display.
[e/a anzeige]		
	Textbasisnr=30000000	Basic number in the file Sprache.txt where the I/O signal texts start.
	Separate_karten_texte=1	1: separate texts for each module 0: the texts of module 1 are used for all modules
[language]		
	Text0=Sprache.txt Text1=SPS.txt	If the entry Language in the [Config] section remains empty, the [language] section is used. Texts of several files are defined. Texts can, for example, be maintained by the MMI developer, others by the PLC programmer and others by the end customer. It must be ensured that the text numbers are strictly assigned.

Section	Term	Description
[SPS_EXEC]		Call of a file that is executable under Windows by a PLC message: Entry=EXE file;parameter;[MAX|MIN|NORM] EXE file: executable Windows file Parameter: transfer parameter for EXE file MAX: maximum window MIN: minimum window NORM: normal window
	entry0=C:\WinNT\Notepad entry1=C:\WinNT\Notepad;Readme.txt entry2=C:\WinNT\Notepad;Readme.txt;Max entry3=C:\WinNT\Notepad;;Min	Examples: See also section SPS_EXEC
[LenzeTools]		
	F1=C:\Programs\lenze\GDC_4_70\bin\gdc32.exe F2=C:\Programs\lenze\ETC\MMI\axcop.exe F3=C:\Programs\lenze\ETC\CoDeSys\CoDeSys.exe F4=C:\Programs\lenze\GDO_1_20\osci.exe F5=c:\Programs\lenze\etc\mmi\etchc1.ht	With KeyCode (F12 diagnostics, F1 axis settings, F2 Lenze tools) the tools keys are activated. The texts for key labelling must be entered in the language file from entry 1501 for F1.
[BARANZ]		Configuration of the bar display
	ANZ1=0;800;10802;1;P30 ANZ2=0;800;10902;2;P31 ANZ3=0;3000;10920;3;4711;DOUBLE Anz4=0;4000;10930;4;D10,WORD	See also section BARANZ following this table.
[pfielddisplay1] [pfielddisplay2] [pfielddisplay3] [pfielddisplay4]		The entries in these sections are evaluated by the HMI when the operating mode is changed. They are not used by default. Each section describes the indexes of the P fields for a specific operating mode, which the NCR is to update cyclically in the area of the ”P field permanent display” of the DPR=s. The assignment of the sections to the operating modes is fixed: 1: Setup 2: Automatic 3: Programming 4: Diagnostics
	P0=1200 . . P31=-1	Example: The contents of the P field 1200 are cyclically updated in the DPR under nc2mmi_r.pfanz_r.val_ad[0]. If the value is -1, there will be no update on this index.
[Zyklen]		(see also: ”Cycle programming”)
	entry0 = L8000; grinding	L8000 is a reference to a section where the cycle is described in more detail. Grinding is a text that is used for cycle selection.
[L8000]		(see also: ”Cycle programming”)
	entry1 = P1200 = 1.23; top entry2 = P1201= 2.34; middle entry3 = P1202= 3.45; bottom	The individual entries describe the cycle.

Section	Term	Description
[WerkzeugVerwaltung]		Determination of the internal tool management (see documentation on the standard HMI).
	Wsk=default.wsk	Name and path of a file with workpiece correction data. After a complete (!) download, this file is transferred to the NCR for presetting.
	wtk=default.wtk	Name and path of a file with tool correction data. After a complete (!) download, this file is transferred to the NCR for presetting.
	magazinplaetze=9	Information on the released S coordinate systems in the magazine management (a maximum of 100 S systems are available at the same time).
	KSxx=a,c-e,g KSyy=e	Information on the relationship between magazine position (a, c-e,..) and corresponding S coordinate system (xx and yy). The entry can be made in the form of an enumeration (a, c, d, e..) or by specifying a range (c-e).
[SPS-Tasten]		In this section, freely configurable ”PLC key” commands can be entered. They can then be called within the MMI software from any softkey menu via a ”hotkey”, which is also configurable. The key information (max. 128) is reported to the PLC via DB1 DW212-219.
	KeyCode=Shift-F8	Determines the key combination for showing and hiding the alternative softkey menu with the PLC key functions defined in the following. A valid entry consists of the specification of a function key (F1 .. F12) and optionally their linkage with any combination of the control keys ”Shift”, ”Alt” and ”Ctrl”.

Section	Term	Description
	KEYxxx=yyyy . . KEYmmm=nnnn,oooo	Determination of the properties of the ”PLC key” to be activated: ● Behaviour as key or switch, ● the sequence of the execution of the functions on the softkeys ● the labelling of the softkeys are achieved by entries of the type KEYxxx= yyyy or KEYmmm=nnnn,oooo. The value ”xxx” describes the index of a bit within the ”virtual keyboard” (**DB1 DW212-DW219**) of the PLC; the value ”yyyy”, ”nnnn” or ”oooo” references the entries in the language file for labelling the corresponding softkeys. If 2 comma-separated language file references are entered (”nnnn,oooo”), the softkey behaves like a switch. Depending on the switch state, the labelling is changed: ”nnnn” stands for the switch state ”off”, ”oooo” for ”on”. The sequence of the entries corresponds to the execution of the PLC key functions on the horizontal softkey line. If more than 8 functions are defined (an entry of the type KEYxxx=-1 defines an empty key), a function for incrementing is offered on the 8th softkey,e.g.: The Ini file contains: Key0 =2800,2927 Key12 =2818,2911 Key13 =-1 Key3 =2812 Key19 =2814 Key29 =2809 Key123 =2801 Key14 =2817 Key22 =2803 Key23 =2805 The language file contains: 00002800,” [0]@C12632256@FArial,8,1,0,3,6” 00002801,” [1]@C12632256@FArial,8,1,0,3,6” : 00002926,”[126]@C12632256@FArial,8,1,0,3,6” 00002927,”[127]@C12632256@FArial,8,1,0,3,6” This results in 2 softkey lines with the following structure: [00] | [18] | | [12] | [14] | [09] | [01] |more.. [17] | [03] | [05] | | | | |more.. After ”[00]” has been actuated, the labelling for softkey 1 changes to [127] [127] | [18] | | [12] | [14] | [09] | [01] |more. Please note that the relationship between the softkey labelling, the set bit in the PLC keyboard and the reaction of the machine is only established by the programmed procedures in the PLC.

Note!

The file can be edited by means of the MMI software (Diagnostics operating mode - MMI-config. function). It must be noted here that some changes are only updated after a restart of the software and some changes are only updated when the control system is reset.

A restart of the software can be achieved by quitting the MMI software and then restarting it. To reset the control system, it must be switched off and on or the Reset pushbutton must be pressed.

SPS_EXEC

In the Delphmmi.ini in the [SPS_EXEC] section, the programs are entered that are to be executed if required. The entry number specifies the index of the program that the PLC must report. The name of the application is entered. Another program parameter can be specified separated by a semicolon.

It can also be specified whether the application is executed in a normal window (NORMAL) or in a maximised or minimised window. The default setting is normal.

```
[SPS_EXEC]
entry0=C:\WinNT\Notepad
entry1=C:\WinNT\Notepad;Readme.txt
entry2=C:\WinNT\Notepad;Readme.txt;Max
entry3=C:\WinNT\Notepad;;Min
```

The PLC now sends a message **SB0_PLC_EXECUTE = $8015;** to the MMI. The first word of the message contains the entry number. If the second data word is non-zero, an error is displayed in the dialogue window.

The HMI always sends back a message to the PLC: **SB0_MMI_EXECUTE = $8016;**. In the first DW, the error number is coded.

0: No error
1: Error executing the program
2: A program is already running
3: Could not find program
4: No program specified for the entry

In the second DW, the number of the requested job is returned.

Note!

Limitation: Only one program from the list can be executed at the same time.

BARANZ

The configuration of the bar displays for Vist and Override can be changed to display a different value from the DPR. Moreover, two different bar displays below the axis positions can be configured in the Automatic operating mode.

For this purpose, there is a new section [BarAnz] in the Delphmmi.ini. This section contains up to four entries for the four bars.

Example:

[BARANZ]
ANZ1=0;800;10802;1;P30
ANZ2=0;800;10902;2;P31
ANZ3=0;3000;10920;3;4711;DOUBLE
Anz4=0;4000;10930;4;D10,WORD

The values of an entry have the following meaning:

1. Min value of the bar or the labelling (currently, the entries < 0 are not supported)
2. Max value of the bar or the labelling (the Max value must be greater than the Min value)
3. Text number of the descriptive text in the language text
4. Bar on which the value is to be displayed:
   1= upper right (standard Vist)
   2= lower right (standard Override)
   3= upper left (standard invisible)
   4= lower left (standard invisible)
5. The value to be displayed; there are three types of specifying a value:
   PXX: Index on the field permanent display; the corresponding field must be entered in the Delphmmi.ini in for the corresponding displays.
   DXX: PLC DB2 display off 0 ... 63,
   XXXX: Direct address in the DPR
6. If the information on the displayed value is specified via DB2 or via a direct address, the value format must also be specified. The following values are permitted:
   BYTE
   DOUBLE
   INTEGER
   LONG
   SINGLE
   WORD

For P fields the data type is always double.

8 PLC programming

8.1 ETC PLC programming with CoDeSys

CoDeSys is a complete development environment for creating and testing PLC programs for the ETC.

CoDeSys offers options for debugging programs similar to modern high level language development systems (setting breakpoints, monitoring variables, recording a trace (oscilloscope function) etc.).

An introduction to PLC programming in accordance with IEC 61131-3 using CoDeSys can be found in the CoDeSys manual (see ”Docu” directory in the CoDeSys installation path).

This user manual is intended as a supplement to the CoDeSys manual. It contains descriptions of functions or adaptations related to CoDeSys and the ETC.

8.2 CoDeSys installation

8.2.1 System requirements for CoDeSys V2.xx

- Pentium processor (Pentium II, 350 MHz or higher recommended)
- 32 MB RAM (64 MB recommended)
- Windows 2000 or XP
- MS Internet Explorer version 4.0 or higher

8.2.2 Installing software

1. Place the CoDeSys setup CD into your CD-ROM drive.
   If the autostart feature for the CD-ROM drive has been enabled in Windows, setup will start automatically.
2. If setup does not start automatically:
   – Select the menu item ”Run” from the start menu.
   – Enter the drive letter of your CD-ROM drive followed by ”setup.exe” (e.g. ”d:\setup.exe”) into the command line and confirm with ”OK”.
3. When selecting the target directory please create a new directory ”CoDeSys” under ”c:\program files\Lenze\ETC”.
4. Follow the instructions of the setup program.
5. Change the installation path to ”c:\program files\lenze\etc\CoDeSys2”.

Stop!

The file ”codesys.ini” in the installation directory may be write-protected after installation from CD. The write-protection must be cancelled, otherwise the CoDeSys settings will not be saved.

8.3 Connecting ETC and PC

The connection to the ETC can be established via one of the following interfaces:

- ► Serial interface
- ► Ethernet interface (only ETCHx, DIN rail design)
- ► DPR interface (only ETCPx, PCI insert card)

8.3.1 V.24 Interface

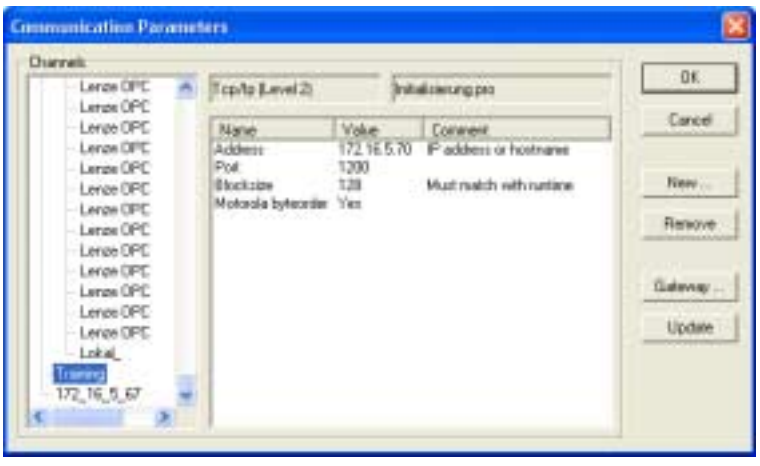
For communication via the serial interface the RS232 or CAN2 connection of the ETC must be connected via a null modem cable to a free COM port of the PC. The driver ”3S Serial RS232 driver” must be selected in the communication parameters of CoDeSys. In the settings for this driver the following values must be selected for the corresponding interface of the control:

ETCconnection	RS232	CAN2
Baud rate	115200	38400
Parity	No	No
Stop bits	1	1
Motorola byte order	Yes	Yes

8.3.2 Ethernet interface (only ETCHx, DIN rail design)

For Ethernet communication the Ethernet connection at the ETCHx must be connected to the PC via an Ethernet cable.

The driver ”3S Tcp/Ip level2 driver” must be selected in the communication parameters of CoDeSys. The following values must be entered in the settings for the driver:

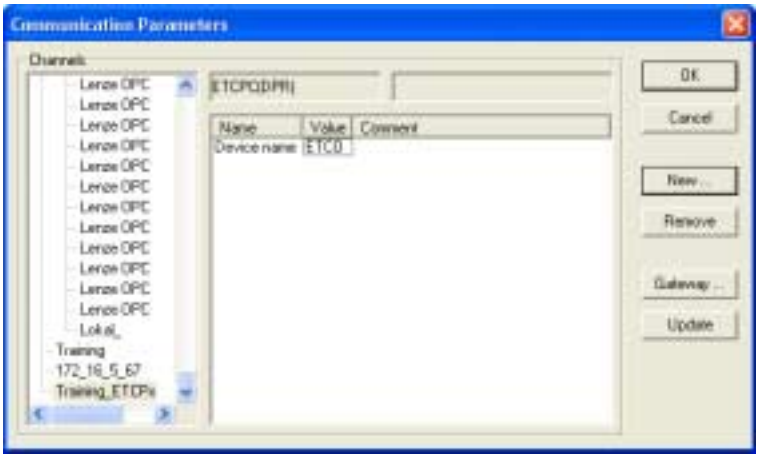


ETC100

8.3.3 DPR interface (only ETCPx, PCI insert card)

For the communication via the DPR interface the Windows WDM driver for the ETCPx and the Lenze MMI Gateway must be installed.

In the communication parameters of CoDeSys the driver ”E*DPR” must be selected. The following values must be entered into the driver settings:



When selecting the driver ”DPR” the controls stated in the MMI Gateway can be selected from the table entry ”Value”.

This is done by double clicking on the corresponding field (here: ETC1) followed by switching via the arrow keys.

8.4 Project planning

8.4.1 Target system setup

When creating a new project in CoDeSys a dialog will automatically open which asks for the hardware used (=target system). After the creation the configuration dialog can be called via the menu item ”target system setup” in the tab ”Resources”.

Note!

The selected target system will be checked during login at the control so that a PLC program cannot be transferred to the wrong target system by CoDeSys.

For a boot project (📖 355) it must be ensured that the selected target system matches the firmware of the control because this will not be checked when loading a boot project e.g. by an ETC MMI.

► Control variant CNC: target system preselect = ETCxC

► Control variant motion: target system preselect = ETCxM

The parameters under ”Target system setup” are preset on the basis of the selected target system and do not normally require modification.

A description of the target system settings can be found in the CoDeSys manual.

8.4.2 Configuring PLC tasks of the ETCxM

Binding PLC tasks

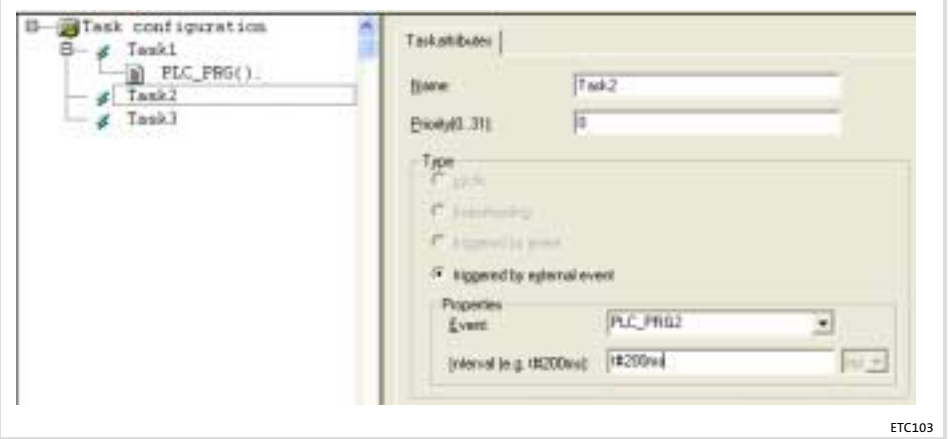
With the ETCxM it is possible to use four PLC tasks independently from each other. Each task can be linked to an IEC program component (PRG) during the call. The program components linked to the tasks are processed almost parallel by the multitasking operating system of the ETCxM .

The processing sequence is a result of the interval time (time-controlled) and the priority of a task. A task will run until a PLC cycle has been completed or another task with higher priority is started. In order for a PLC program to be processed by the ETCxM at least one task must have been created in the task configuration of CoDeSys and linked to one IEC program component.

ETC102

Task properties

The properties of the individual tasks are configured in the ”Task configuration” of CoDeSys.



The following points must be noted:

- The greater the value for priority, the greater the task priority.
- A task with an interval of ”t#0s” must have a priority of no more than 10, otherwise the control will set the priority of all tasks to 0 and reports an error (module 13, error number 202).
- In addition a task with an interval of ”t#0s” must have the lowest priority of all tasks, otherwise the control will set the priority of all tasks to 0 and reports an error (module 13, error number 201). The tasks with a lower priority than the tasks mentioned above would then not be executed.
- If there are several tasks with an interval ”t#0s”, these tasks must all have the same priority.
- The interval time must be selected between ”t#0ms” (constantly running) and ”t#1000ms”.
- When making the entry for ”Event” in the task properties of CoDeSys each task must be linked to an event variable.
- The operating system of the ETCxM then allocates the 4 tasks to the corresponding call of the CoDeSys program component linked to the task. Each task has an event variable allocated to it.

Task number	Event variable
1	PLC_PRG1
2	PLC_PRG2
3	PLC_PRG3
4	PLC_PRG4

Stop!

When accessing data or calling components which are used in several tasks it must be noted that no synchronisation exists between the tasks.

The PLC programmer must ensure that this does not cause any problems.

Process image

When allocating the CoDeSys program components to the PLC tasks it must be noted that each task is only allocated to a part of the process image. The process image is a copy of the current states of the input/output blocks (hardware) to which the PLC has access.

Before every PLC cycle (complete program run) the current states of the input modules are copied into the process image and after each PLC cycle the current states of the outputs in the process image are transferred to the output modules.

Because each task is allocated to a part of the process image, only the corresponding part of the process image will be updated before and after the program component to which a task is allocated will be run.

8.4.3 Configuring PLC tasks of the ETCxC

Binding PLC tasks

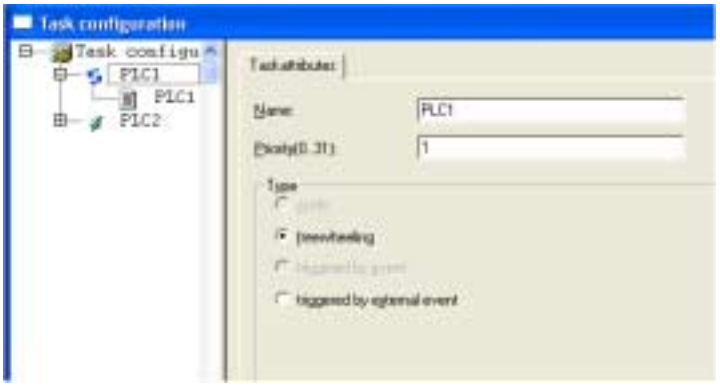
Unlike the ETCxM, the ETCxC only has 2 tasks, with task 1 being the main task. The 2nd (lower priority) task is used as an interface to control-internal signals for operation (e.g. error management).

In order to run a user-specific component (IEC program component: PROGRAM) within the task instead of control-internal functions, the following steps are required:

Step 1:

Two tasks must be inserted into the task management of CoDeSys:

1st task entry:



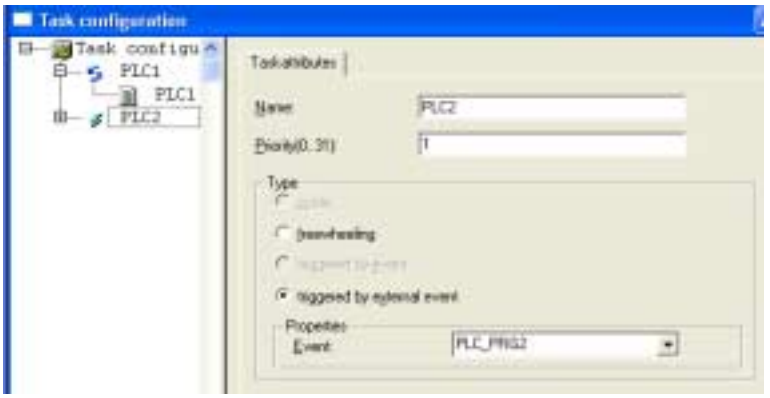
ETC104

Step 2:

As identification for the control that this task entry is the 2nd (lower priority) task ”PLC_PRG2” must be entered into the field ”event” (or OB20 if the program component to be linked to the task was called ”PLC_PRG2”).

2nd task entry:

As soon as the task management is used by CoDeSys, a task entry for the main component ”PLC_PRG” must also be entered, otherwise this component will no longer be called:



If only the main task is required, it is not necessary to edit the task configuration.

Step 3:

Link the tasks to the IEC program components.

Any IEC program component can be linked to the 1st task entry. Because the component designation ”PLC_PRG” has a special meaning it should be renamed (e.g. ”PLC_PRG1”) and then linked to the 2nd task entry.

Note!

When accessing data or when calling components which are used in both tasks (PLC_TASK (PLC_PRG1), e.g. Name: BDT_TASK (PLC_PRG2)) it must be ensured that no synchronisation between the two tasks exists.

The PLC programmer must ensure that this does not cause any problems.

8.4.4 Configuring I/O modules

CoDeSys includes a CAN Bus configurator. The configurator is based on so-called EDS files (EDS = Electronic Data Sheet), which must be stored in the subdirectory ”ETC” of the target directory.

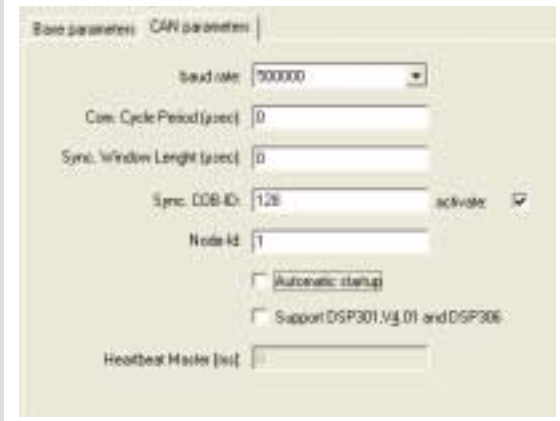
A general description of the configuration of a CAN module can be found in the CoDeSys documentation.

8.4.4.1 Configuring CAN Master (global CAN settings)

Basic parameters

Automatic address: The switch should be disabled, otherwise CoDeSys will allocate the addresses automatically.

CAN parameters



ETC106

Baud rate: The baud rate for the CAN Bus must be selected in accordance with the settings of the connected CAN modules.

Note!

Entering the baud rate is only enabled for the ETCxM.

For the ETCxC this parameter is without function. Only settings of the machine constants are possible.

Com. Cycle Period: Will not be evaluated by the control.

Sync. Window Length: Will not be evaluated by the control.

Sync COB-ID: Will not be evaluated by the control.

Node-ID: Enter a ”0” here.

Automatic start up: Will not be evaluated by the control.

Support DSP301 ...: Will not be evaluated by the control.

Heartbeat Master: Will not be evaluated by the control.

8.4.4.2 CAN slave configuration

Basic parameters

Module ID: ID number of the module.

Node ID: The node ID is entered in the CAN parameter tab.

ETCxM Input/Output address: The basic addresses must always be stated as word addresses. The address defines where the process data of the module are located within the process image (I/O memory). This also determines to which PLC task the data will be allocated. The following table shows the allocation of the process image to the individual PLC tasks:

Task no.	I/O range	
	from	to
1	%W0	%W1023
2	%W1024	%W1535
3	%W1536	%W1791
4	%W1792	%W2047

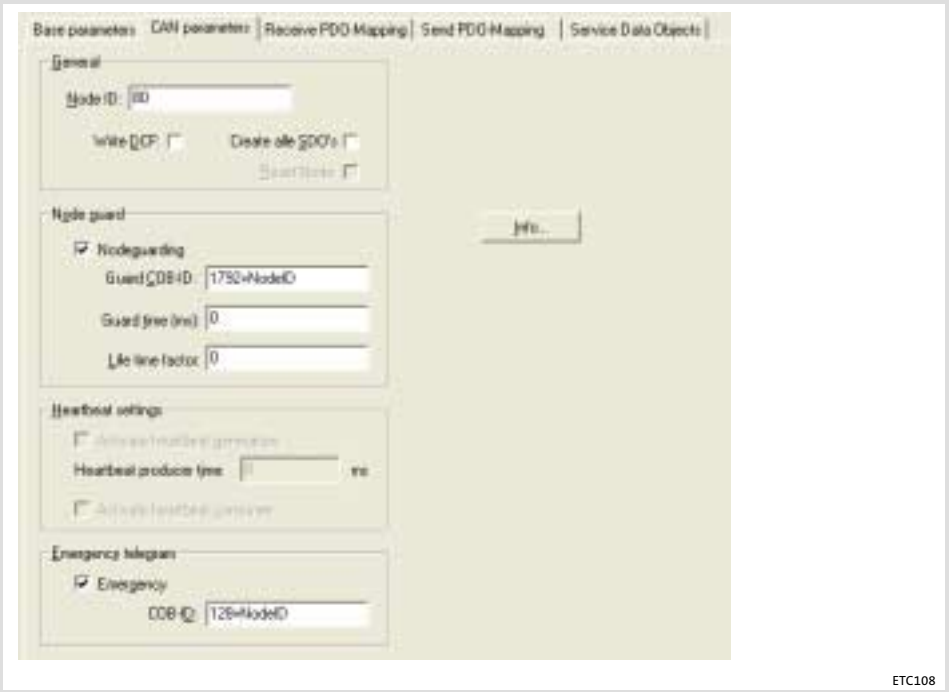
Further information can be found in the chapter "Addressing" (📖 349).

ETCxC Input/Output address: The basic addresses must be issued as word addresses within the range of %QW64 / %IW64 to %QW127 / %IW127.

Further information can be found in the chapter "Addressing" (📖 349).

Diagnostic address: This field will not be evaluated by the control.

CAN parameters



General: In node ID the CAN ID (node number) set for the module must be entered.

Nodeguarding: Nodeguarding is used to detect whether a CANopen module is connected to the Bus. To do so the control sends a message to the module in ”guard time” intervals and waits for a response.

If the control does not receive a response after the ”Life time factor” requests, an error message is generated.

If no ”Guard Time” and no ”Life Time Factor” have been defined the CAN module is not monitored; this means that a failure will go unnoticed.

Heartbeat settings: Will not be evaluated by the control.

Emergency telegram: If this entry is selected, the control ignores ”emergency telegrams” from the control.

PDO mapping

Via PDO (Process Data Object) the process data are transferred at CANopen; this means the states of the digital and analogue inputs/outputs.

Each PDO has a unique COB ID (385-1407) allocated to it, with the COB IDs for the CANopen IO modules being issued as follows, assuming a maximum of 127 modules:

1. Tx-PDO: 384 + NodeID
1. Rx-PDO: 512 + NodeID
2. Tx-PDO: 640 + NodeID
2. Rx-PDO: 768 + NodeID
3. Tx-PDO: 896 + NodeID
3. Rx-PDO: 1024 + NodeID
4. Tx-PDO: 1152 + NodeID
4. Rx-PDO: 1280 + NodeID

(Tx = Transmit, Rx = Receive as seen from the perspective of the CANopen module).

This means that without limitation of the maximum number of modules 4 receive and transmit PDOs can be issued per module. If in deviation from the standard more PDOs are to be defined per module, it must be ensured that the COB ID's will not be issued twice.

The first Tx/Rx PDO contains the digital data and the second Tx/Rx PDO the analogue data. For the second PDO "asynchronous-only RTR" should always be entered under "Transmission Type" since this is the standard setting for most modules when transferring analogue values.

Service Data Objects

These entries are not evaluated by the control.

8.4.4.3 Checking the number of inputs and outputs

The control carries out an implicit check of the number of digital/analogue inputs and outputs.

Here the assumption is made that the digital data are located in the first Tx/Rx PDO and the analogue values in the second Tx/Rx PDO. This assumption and the resulting check will only take place if in the EDS file for the CANopen module for the object entry 1000_{hex} (Device Profil Number) the IO support for digital and analogue inputs/outputs has been selected in the additional information (highest value 16-bit) (see CiA DS401: Device Profil for Generic I/O Modules).

Note!

The control generates an error message if the number of configured digital/analogue inputs/outputs of a PDO exceeds the actual number.

Extract from an EDS file

```
[1000]
ParameterName=DeviceType
ObjectType=0x7
DataType=0x7
AccessType=ro
PDOMapping=0
DefaultValue=0x30191
```

Description of object index 1000hex

Additional information				General information	
Special functionality		I/O functionality		Device Profile Number	
31	24	23	16	15	0

I/O functionality

Bit 16: Digital inputs
Bit 17: Digital outputs
Bit 18: Analogue inputs
Bit 19: Analogue outputs

Disable check

To disable the check of the number of IOs the DefaultValue = 0x191 has to be set. If a module also transfers analogue data with the 1st Tx/Rx PDO, the DefaultValue = 0xC0191 must be set, i.e. the support for digital I/Os must be disabled.

8.4.4.4 Searching for configured CANopen modules

The control searches for all modules configured in the control configuration before starting the actual communication (via PDO) with the module.

Searching for the modules is done via NodeGuarding, if configured, otherwise via SDO Transfer (reading of Object index 1000_{hex}).

Disable search

For CANopen modules which do not support NodeGuarding or SDO Transfer, searching for modules can be disabled. For this a 0 must be entered in the EDS file for the CANopen module for the object entry 1000_{hex} (Device Profil Number).

8.4.5 Addressing

The ETC is based on the big-endian data model, i.e. all data types are in the memory with those of the highest value byte, i.e. the ”bigger end”, on top. This is especially important when communicating with a PC-HMI (e.g. ETC-MMI) via DB2 (ETCxC), because PCs are based on the little-endian data model. The control provides a number of functions to simplify this (📖 393).

The direct representation of individual memory cells is achieved using special rows of signs. They are made up of a percentage sign ”%”, an area prefix, a prefix for the data width and two or three natural numbers separated by dots.

Area prefix	Meaning
I	Input
Q	Output
M	Data block (DBO ... DB15) of the ETC

Data width prefix	Meaning
X	1 Bit
B	Byte (8 Bits)
W	Word (16 Bits)
D	Double word (32 Bits)

8.4.5.1 Addressing I/O modules

The I/O addressing is always organised at the word level. The addresses of I/Os derive from the internal organisation of the process image of the control.

Access type	Syntax	Comment
In bits	%IX x.y / %QX x.y	x: number of the data word (0 ... 127) y: Bit in the word (0 ... 15)
In bytes	%IB x / %QB x	x: number of the byte (0 ... 254)
In words	%IW x / %QW x	x: Number of the data word
In double words	%ID x / %QD x	x: Number of the double word

Stop!

Access by word: Because of the internal data organisation it must be noted for access by byte that the high byte is placed first in the memory followed by the low byte (see example).

Double word access: Because of the internal data organisation there is normally no point in a double word access, because high and low words would be transposed (see example).

8.4.5.2 Addressing data blocks

The data blocks (DB0 - DB15) are addressed via the area prefix ”M”.

Access type	Syntax	Comment
In bits	%MX x.y.z	x: number of the data block y: word in the data block (0 ... 255) z: bit in the word (0 ... 15)
In bytes	%MB x.y.z	x: number of the data block y: word in the data block (0 ... 255) z: left (higher value) or right (lower value) byte of the word (left = 1, right = 0)
In words	%MW x.y	x: number of the data block y: word in the data block (0 ... 255)
In double words	%MD x.y	x: number of the data block y: double word in the data block (0 ... 127)

Stop!

Double word access: Because of the internal data organisation there is normally no point in a double word access, because high and low words would be transposed (see example).

8.4.6 Remanent variables

With the key word ”RETAIN” remanent data can be defined for the variable declaration in CoDeSys (see CoDeSys documentation). A total of 12 kB are available for remanent data.

8.4.7 Object directory (parameter manager)

It is possible in CoDeSys to create a so-called CANopen object directory which can be read and written to by a CANopen device (e.g. CANopen control terminal).

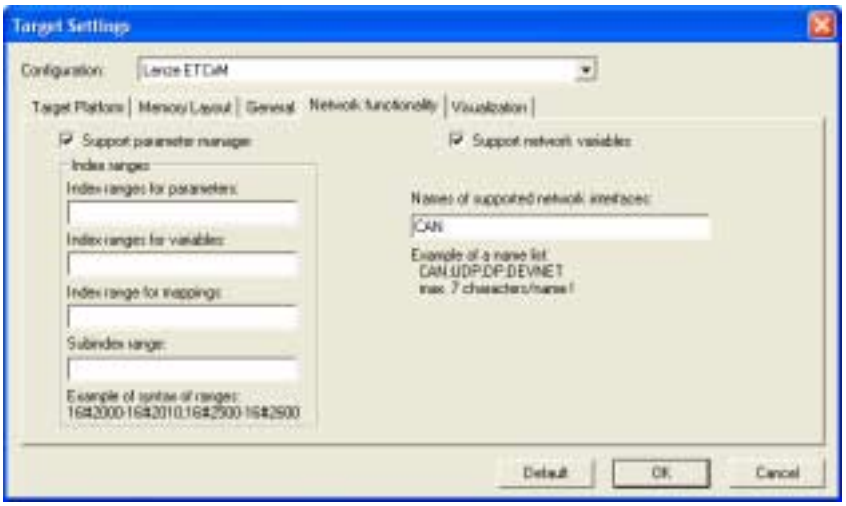
To enable the transfer of an object directory via the CAN Bus the object directory must be announced to the control via the function ”InitServerSdo()” (see library ”ServerSdo.lib”).

Note!

When inserting the library ”ServerSdo.lib” into the library management of CoDeSys additional libraries (SysExtensions.lib, SysLibCallback.lib, Netvarcan_Lib.lib) will be inserted automatically.

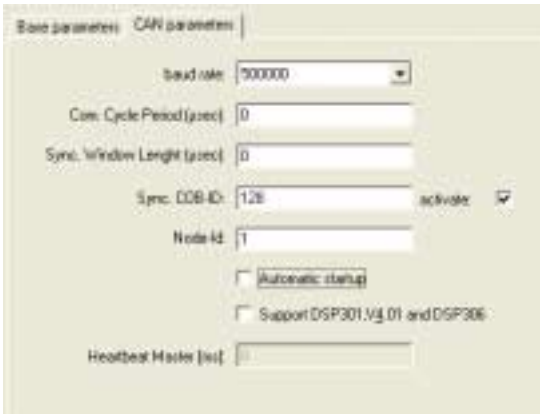
An object directory can be created via the object directory editor of CoDeSys. To be able to use the object directory editor the support for object directories and network variables must be enabled in CoDeSys under **TargetSystemSettings -> Network functionality** (see figure below).

Define the index or subindex range for SDO parameters and variables and enter ”CAN” as the name of supported network interfaces.



ETC109

The CANopen address of the control must be defined under **Control configuration -> CAN Master -> CAN Parameter->Node ID**.



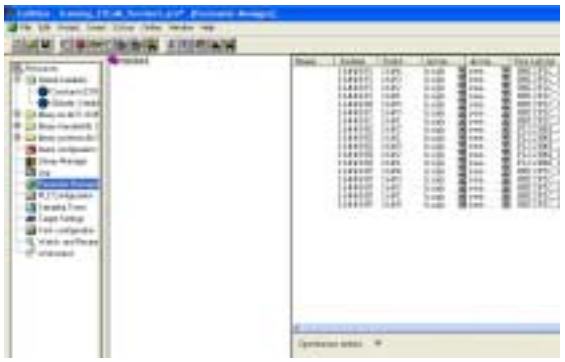
ETC116

When defining the Node ID the following must be noted:

If the node ID is in the range 1 ... 63, two server SDO channels with the addresses ”Node ID” and ”Node ID + 64” will be created, i.e. the control receives 2 CANopen addresses. Because the use of 2 SDO channels is not normally required, a node ID in the range 64 ... 127 should be issued, so that only one SDO channel will be created.

In the object directory editor of CoDeSys entries must only be made in the tab ”Variable”.

Create a new object directory. In it, define the variables to be exchanged with other controls.



ETC110

The tab Variable must have been selected. The indexes can then be entered with their variable names.

Index: In hexadecimal notation

Subindex: The range of the subindexes is defined in the target settings.

Access: Access frequency

Attribute: Attribute of the variables (write, read)

Variable: Name of the variables (Caution: the text must be preceded by a dot.)

8.5 Network variables

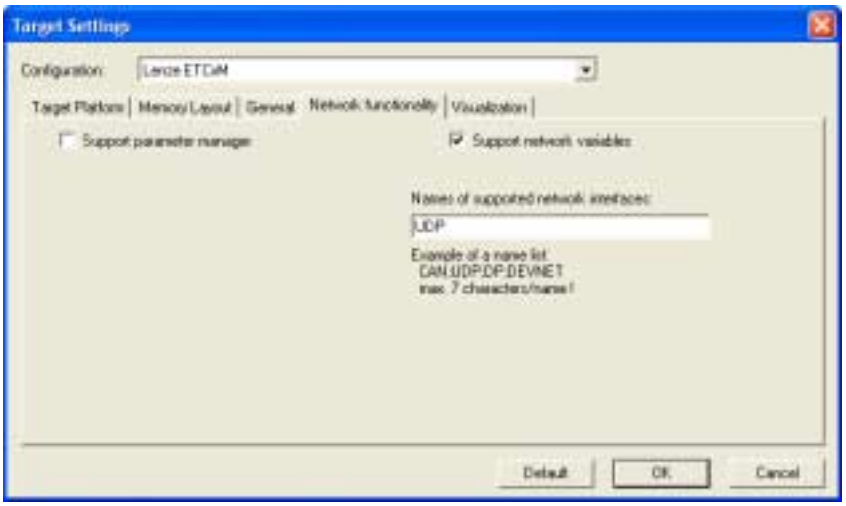
Network variables are a way of transferring data between two or several controls. Currently network variables are implemented on the basis of UDP. The variable values are transferred automatically on the basis of broadcast messages. These services are not confirmed by the protocol, i.e. there is no control whether the message actually reaches the recipient. The network variable transfer corresponds more to a 1 (sender) to n (recipient) connection.

Condition

The libraries ”NetVarUdp_LIB.lib”, ”SysLibCallback.lib” and ”SysLibSocket.lib” have been inserted into the library management of CoDeSys.

8.5.1 Settings in the target system

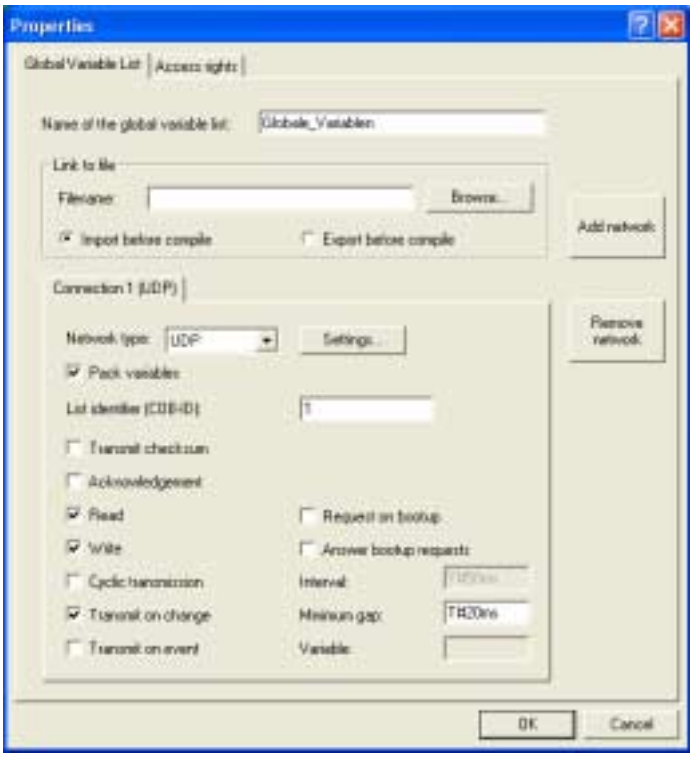
Enable the check box ”Support network variables” in the tab ”Network functionality” of the dialog box ”Target settings”. Enter ”UDP” under ”Names of supported network interfaces”.



ETC111

8.5.2 Settings in the global variable list

Create a new global list of variables. Here you define the variables to be exchanged with other controls. The transfer properties can be defined via the properties dialog of the variable list.



ETC112

You can define the network properties of this variable list by pressing the button **Add network connection**. The options have the following meaning:

Network type: As **network type** UDP must be selected.

Read: Read means that the variable values of one (or several) other controls will be read.

Write: Write means that the variables of this variable list or sent to other controls. It is recommended to select only one option for each variable list, i.e. either only read or only write. If different variables of a project are to be read and written, use several variable lists: one for reading and one for writing. It is also recommended that only one control per network will send the same variable list.

Cyclical transmission: Only valid if "Write" has been enabled. The values will be sent in the time intervals defined, irrespective of whether they have changed.

Transmit on change: The values will only be sent if the value has changed or if the time defined for the minimum interval has expired.

Pack variables: If this option is enabled the variables will be combined to a transfer unit where possible. For UDP a transfer unit has a size of 256 bytes. If not all variables of the list fit into one transfer unit, several transfer units will be created for this list. If the option is disabled, each variable will go into its own transfer unit.

If ”transfer when modified” has been configured a separate check takes place for each transfer unit whether it has changed and must be sent.

List identifier: The list identifier is used as a unique ID to exchange variable lists between different projects. Variable lists with the same list identifier will be exchanged.

It must be ensured that the definitions of the variable lists with the same list identifier match in the different projects.

For this the feature ”File link” can be used. The variable list is exported from one project. The other projects import it so that the content does not have to be re-entered.

To ensure the correct data exchange between controls the global variable lists in both projects must match. One project can export the file prior to the transfer, the others should import it prior to the transfer.

Besides simple data types a variable list can also contain structures and arrays. The elements of these combined data types will be sent individually. If a variable list is larger than a transfer unit, the data are divided into several transfer units. It can, therefore, not be guaranteed that all data of the variable list will be transferred within a single cycle.

Parts of the variable list can be received in different cycles. This can also be the case for variables containing structure and array types.

8.6 Generate program

After creating a project in CoDeSys in the programming languages of IEC61131-3 the project must be translated into a program which can be executed by the control.

Via the menu item **Online -> Login** an executable program is automatically created from the project currently being edited and transferred into the RAM of the control.

If there is an online connection to the control it is possible via the menu item **Online -> Create boot project** to transfer the current project into the Flash PROM of the control. In addition a file ”Projectname.prg” will be created in the project directory on the hard drive if in **Project -> Options -> Icon configuration** the control field ”Create binary file for the application” has been selected.

This file can also be loaded into the control via the monitor interface of the control using Z modem transfer. In this case the file name of the boot project must be changed to ”default.prg”. For the ETCxC this file can also be transferred to the control via the ETC-MMI.

8.7 Interface to the ETC

8.7.1 Data blocks

The integrated PLC has an internal RAM range available which is divided into 16 so-called data blocks. Each data block (DB) contains 256 data words (DW) of 16 Bit.

8.7.2 System variables of the ETCxC

The system variables represent a predefined number of variables from the data blocks.

In CoDeSys the system variables can be inserted into a project via the menu item **Insert→ operand → System variables** or the input assistance (via function key ”F2”).

the following is a list of the defined system variables:

Data block 0

DB0_sps2nc_taste_an_funktion_w	%MW0.0	
DB0_sps2nc_nc_fehlerquittung_w	%MW0.1	
DB0_nc2sps_fehlertext_s	%MW0.16	
DB0_nc2sps_fehlerinfo_s	%MW0.96	
DB0_nc2sps_aktuelles_menu_w	%MW0.128	
DB0_nc2sps_ob20_ereignis_w	%MW0.129	
DB0_nc2sps_einzeltastenmeldung_w	%MW0.130	
DB0_nc2sps_tastennummer_b	%MB0.130.0	
DB0_nc2sps_tastenzustand_b	%MB0.130.1	
DB0_nc2sps_eingabeabschluss_w	%MW0.131	
DB0_nc2sps_funktionsstatus_w	%MW0.132	
DB0_nc2sps_nc_fehlerzaehler_w	%MW0.133	
DB0_nc2sps_nc_fehlerklasse_w	%MW0.134	
DB0_nc2sps_nc_fehlermodul_w	%MW0.135	
DB0_nc2sps_nc_fehlernummer_w	%MW0.136	
DB0_nc2sps_aktuelles_untermenu_w	%MW0.137	
DB0_nc2sps_g253_textausgabe_s	%MW0.148	

Data block 1

DB1_sps2nc_notaus_bit	%MX1.0.0	
DB1_sps2nc_vorschubfreigabe_bit	%MX1.0.1	
DB1_sps2nc_soforthalt_bit	%MX1.0.2	
DB1_sps2nc_vorschubfreigabe_w	%MW1.1	
DB1_sps2nc_position_halt_w	%MW1.2	
DB1_sps2nc_verfahrtastenfreigabe_plus_w	%MW1.3	
DB1_sps2nc_verfahrtastenfreigabe_minus_w	%MW1.4	
DB1_sps2nc_reglerfreigabe_w	%MW1.5	
DB1_sps2nc_einlesefreigabe_bit:	%MX1.7.0	
DB1_sps2nc_programmstart_b	%MB1.9.0	
DB1_sps2nc_use_spsoverride_b	%MB1.9.1	
DB1_sps2nc_spindel_ein_aus_b	%MB1.10.0	
DB1_sps2nc_spindel_richtung_b	%MB1.11.0	

DB1_sps2nc_programmstop_b	%MB1.12.0	
DB1_sps2nc_unterbrechen_bit	%MX1.12.8	
DB1_sps2nc_einzel_folgesatz_bit	%MX1.13.0	
DB1_sps2nc_satzausblenden_bit	%MX1.13.8	
DB1_sps2nc_rueckzug_bit	%MX1.14.0	
DB1_sps2nc_tastensignale_aw	%MW1.16	ARRAY[0..1] OF WORD
DB1_sps2nc_qbit_signale_aw	%MW1.18	ARRAY[0..3] OF WORD
DB1_sps2nc_freigaben_mmi_w	%MW1.31	
DB1_sps2nc_mfkt_quitt_bit	%MX1.32.0	
DB1_sps2nc_sfkt_quitt_bit	%MX1.33.0	
DB1_sps2nc_hfkt_quitt_bit	%MX1.33.0	
DB1_sps2nc_tfkt_quitt_bit	%MX1.34.0	
DB1_sps2nc_extsync_enable_bit	%MX1.35.0	
DB1_sps2nc_programmhalt_aktiv_bit	%MX1.35.8	
DB1_sps2nc_qin_mask_aw	%MW1.36	ARRAY[0..1] OF WORD
DB1_sps2nc_qout_mask_aw	%MW1.38	ARRAY[0..1] OF WORD
DB1_sps2nc_qout_akt_aw	%MW1.40	ARRAY[0..1] OF WORD
DB1_sps2nc_qin_offset_ab	%MW1.42	ARRAY[0..1] OF BYTE
DB1_sps2nc_qout_offset_ab	%MW1.43	ARRAY[0..1] OF BYTE
DB1_sps2nc_spsoverride_allg_w	%MW1.80	
DB1_sps2nc_spsoverride_spindel_w	%MW1.81	
DB1_sps2nc_spsoverride_ozillation_w	%MW1.82	
DB1_sps2nc_spsoverride_sps_w	%MW1.83	
DB1_sps2nc_verfahr_ab	%MB1.84.0	ARRAY[0..11] OF BYTE
DB1_nc2sps_betriebsbereit_1_bit	%MX1.128.0	
DB1_nc2sps_betriebsbereit_2_bit	%MX1.128.1	
DB1_nc2sps_referenzpunkt_w	%MW1.129	
DB1_nc2sps_referenzfahrt_laeuft_w	%MW1.130	
DB1_nc2sps_betriebsart_w	%MW1.131	
DB1_nc2sps_nc_programm_laeuft_bit	%MX1.132.0	
DB1_nc2sps_nc_programm_laeuft_1_bit	%MX1.132.1	
DB1_nc2sps_satzvorlauf_aktiv_bit	%MX1.132.2	
DB1_nc2sps_einzelsatz_laeuft_bit:	%MX1.132.3	
DB1_nc2sps_unterbrechen_aktiv_bit	%MX1.132.4	
DB1_nc2sps_alle_achsen_stehen_bit	%MX1.132.8	
DB1_nc2sps_wahlweise_halt_b	%MB1.133.0	
DB1_nc2sps_programmstartzaehler_b	%MB1.133.1	
DB1_nc2sps_achse_faehrt_w	%MW1.134	
DB1_nc2sps_modal_am_ziel_w	%MW1.135	
DB1_nc2sps_spindel_aktiv_w	%MW1.136	
DB1_nc2sps_programmstart_b	%MB1.137.0	
DB1_nc2sps_canmodul_vorhanden_w	%MW1.138	
DB1_nc2sps_can_status_w	%MW1.139	
DB1_nc2sps_programmstop_b	%MB1.142.0	
DB1_nc2sps_vorschub_halt_b	%MB1.144.0	
DB1_nc2sps_mfkt_strobe_bit	%MX1.150.0	
DB1_nc2sps_mfkt_w	%MW1.151	
DB1_nc2sps_hfkt_strobe_bit	%MX1.159.0	
DB1_nc2sps_hfkt_w	%MW1.160	

DB1_nc2sps_sfkt_strobe_bit	%MX1.159.0	
DB1_nc2sps_sfkt_w	%MW1.160	
DB1_nc2sps_tfkt_strobe_bit	%MX1.168.0	
DB1_nc2sps_tfkt_w	%MW1.169	
DB1_nc2sps_refpunkt_angefahren_w	%MW1.177	
DB1_nc2sps_endschalter_plus_w	%MW1.178	
DB1_nc2sps_endschalter_minus_w	%MW1.179	
DB1_nc2sps_referenznocken_w	%MW1.180	
DB1_nc2sps_reserveeingang_w	%MW1.181	
DB1_nc2sps_slave_endschalter_plus_w	%MW1.182	
DB1_nc2sps_slave_endschalter_minus_w	%MW1.183	
DB1_nc2sps_slave_referenznocken_w	%MW1.184	
DB1_nc2sps_slave_reserveeingang_w	%MW1.185	
DB1_nc2sps_copmodul_vorhanden_aw	%MW1.192	ARRAY[0..7] OF WORD
DB1_nc2sps_spsoverride_allg_w	%MW1.200	
DB1_nc2sps_spsoverride_spindel_w	%MW1.201	
DB1_nc2sps_spsoverride_ozillation_w	%MW1.202	
DB1_nc2sps_spsoverride_sps_w		%MW1.203
DB1_nc2sps_verfahr_ab	%MB1.204.0	ARRAY[0..11] OF BYTE
DB1_nc2sps_mmi_tasten_aw	%MW1.212	ARRAY[0..7] OF WORD

Data block 2

DB2_sps2mmi_zustand_aw	%MW2.0	ARRAY[0..15] OF WORD
DB2_sps2mmi_anzeige_aw	%MW2.16	ARRAY[0..63] OF WORD
DB2_sps2mmi_hinweis_aw	%MW2.80	ARRAY[0..15] OF WORD
DB2_sps2mmi_quit_cnt_w	%MW2.96	
DB2_sps2mmi_msg_cnt1_w	%MW2.97	
DB2_sps2mmi_msg_auftrag_w	%MW2.98	
DB2_sps2mmi_msg_laenge_w	%MW2.99	
DB2_sps2mmi_msg_daten_aw	%MW2.100	ARRAY[0..25] OF WORD
DB2_sps2mmi_msg_cnt2_w	%MW2.126	
DB2_sps2mmi_ok_cnt_w	%MW2.127	
DB2_mmi2sps_zustand_aw	%MW2.128	ARRAY[0..15] OF WORD
DB2_mmi2sps_tasten_aw	%MW2.144	ARRAY[0..15] OF WORD
DB2_mmi2sps_quit_cnt_w	%MW2.160	
DB2_mmi2sps_msg_cnt1_w	%MW2.161	
DB2_mmi2sps_msg_auftrag_w	%MW2.162	
DB2_mmi2sps_msg_laenge_w	%MW2.163	
DB2_mmi2sps_msg_daten_aw	%MW2.164	ARRAY[0..25] OF WORD
DB2_mmi2sps_msg_cnt2_w	%MW2.190	
DB2_mmi2sps_ok_cnt_w	%MW2.191	
DB2_nc2sps_technodaten_aw	%MW2.192	ARRAY[0..31] OF WORD
DB2_nc2sps_maschinenkonstante_aw	%MW2.224	ARRAY[0..31] OF WORD

Data block 13

DB13_nc2mmi_achskennung_ab	%MW13.12	ARRAY[0..15] OF BYTE

8.7.3 System variables of the ETCxM

Currently the data blocks of the PLC are freely available as data memory for the ETCxM.

8.7.4 Using machine constants in the ETCxC

In the ETCxC the machine constants **MK_DW224_255 [MC_DW224_255]** are freely available to store machine-specific values.

For the ETCxC these are shown in the data block 2 after data word 224, so that the content can be read by the PLC. The system variable ”DB2_nc2sps_maschinenkonstante_aw” [”DB2_nc2plc_machineconstant_aw”] points to the values of these machine constants.

The machine constants **MK_TECHNOLOGY_DATA_1** to **MK_TECHNOLOGY_DATA_4** are available for technology-dependent data. The contents of these machine constants are stored in the parameter field.

Index	Machine constant
760 ... 769	MK_TECHNOLOGY_DATA_1
770 ... 779	MK_TECHNOLOGY_DATA_2
780 ... 789	MK_TECHNOLOGY_DATA_3
790 ... 799	MK_TECHNOLOGY_DATA_4

These parameter field entries can be read via the functions **READ_PARAM_INT**, **READ_PARAM_DINT** etc.

8.7.5 Using machine constants in the ETCxM

The machine constants **MK_TECHNOLOGY_DATA_1** to **MK_TECHNOLOGY_DATA_4** are available for technology-dependent data.

These machine constants contain 10 entries each. The value range of each entry corresponds to a 64 bit floating point value (IEC61131 data type LREAL). The individual entries can be read via the function ”GetUserParam()” from the PLC.

8.7.6 Parameter field of the ETCxC

The ETCxC provides a so-called parameter field (P field) which contains different system parameters which might be of interest to the PLC.

In ”SysEtc.LIB” functions are available for the read and write access to parameter entries (see ”READ_PARAM_INT” or ”WRITE_PARAM_INT” etc.).

The individual entries in the parameter field are described in the chapter ”ETC programming NC operating system”.

8.7.7 Operating data of the ETCxC

The control features an operating data field where so far only 2 entries from the control are being used. The entries are accessible via the functions READ_SYSPARAM and WRITE_SYSPARAM.

Index	Description
101	Cycle time of a second task (PLC_PRG2) [10 ms]
102	Cycle time of the PLC (PLC_PRG) [ms]

Note!

The cycle time of the PLC must be equal to the rough interpolation cycle (see also machine constant MK_DELTAT).

8.7.8 Reading error messages for the ETCxM

Active error messages can be read for the ETCxM via functional components of the library ”mc.lib”. A difference is made between axis-specific errors and operating system errors. There is one function for reading and one for acknowledging (resetting) an existing error for each.

Error source	PLC components
Axes	MC_ReadAxisError MC_Reset
System	MC_ReadSysError MC_ResetSysError

8.8 Library

The functions and functional blocks of the two axis representations ETCxC and ETCxM are the same barring a few exceptions. Nonetheless there is an ETC system library for each variant:

ETCxC: SysEtc.LIB

ETCxM: SysETCxM.LIB

All functions are described in detail below. They are found in both libraries. Special functions of the variant are highlighted.

8.8.1 General functions

8.8.1.1 CenterString

Declaration

```
FUNCTION CenterString: INT
VAR_INPUT
    string_s        : STRING(255)       (* String, which is to be reformatted *)
    linelen_di      : DINT              (* line length for centering *)
END_VAR
```

Description

The function adds spaces to the string for centering. The return value corresponds to the resulting total length of the string.

Ensure that the string variable declared is large enough to accommodate the resulting string!

Example

```
test_s              : STRING (20):= 'Hello';
CenterString(test_s, 20);
```

Invoking CenterString will insert 7 spaces before "Hello".

8.8.1.2 FMOD

Declaration

```
FUNCTION FMOD: LREAL
VAR_INPUT
    x: LREAL;
    y: LREAL;
END_VAR
```

Description

The function returns the floating point Rest of Division x/y (with the prefix of x).

Example

```
Result_lr           : LREAL;
Result_lr           := FMOD(10.5, 3.2); Result_lr = 0.9
```

8.8.1.3 Format

Declaration

```
FUNCTION FORMAT: INT
VAR_INPUT
    string_s        : STRING(255);      (* target string *)
    FORMAT_S        : STRING(80);       (* Format string *)
    PARAMETER_P     : DINT;             (* Address of a structure or variable *)
END_VAR
```

Description

All characters in the format string which are not part of the format definition will be copied into STRING_S.

Format string: The % sign in a format string opens a format definition with the general form:

% Flag output field lengths Accuracy Data type declaration

Flag	
<nothing>	flush right, leading spaces / nulls
-	flush left, following spaces / nulls
+	Always output operational sign

Output field length	
0n	min. n digits, fill with nulls
n	min. n digits, fill with spaces

Accuracy	
<nothing>	6 digits
.0	do not output a decimal point
.n	output max. n digits after the decimal point

Data type (CoDeSys)	Data type definition (format string)
DWORD	u
DINT	D
LREAL	lf
STRING	s

The returned value corresponds to the number of characters which have been copied to the string STRING_S.

Example

```
TYPE TEST_R
STRUCT
    Dw          : DWORD;
    l           : LREAL;
END_STRUCT
END_TYPE

    s_s         : STRING(80);
    ret_i       : INT;
    di          : DINT;
    t_r         : TEST_R;
```

One parameter:

```
    di          : 345;
    ret_i       : = FORMAT(s_s, '%d', ADR(di)); ret_i = 4; s_s = >-345'
```

Several parameters:

```
    t_r.dw      : = 123;
    t_r.lr      : = 4.321;
    ret_i       : = FORMAT(s_s, '1: %u, 2: %f', ADR(t_r)); ret_i = 19; s_s = '1: 123, 2:
                  4.321000'
```

8.8.1.4 GetFirmwareVersion

Declaration

```
FUNCTION GetFirmwareVersion: STRING(80)
VAR_INPUT
    type_i      : INT;
END_VAR
```

Parameters

type_i	Version string
1	NCR
2	Bootloader

Description

The function returns version strings.

8.8.1.5 GetMacAddr (nur ETCxM)

Declaration

```
FUNCTION GetMacAddr: BOOL
VAR_INPUT
    pMac      : DINT;        (* Address of an array of the teyp ARRAY[0..5] OF BYTE *)

END_VAR
```

Description

With this function the MAC address of the Ethernet controller of the control can be read. The function must be passed the address of a memory area of 6 bytes (see example). The return value of the function indicates whether the address could be read.

Example

```
ret_bit          : BOOL;
mac_a            : ARRAY[0..5] OF BYTE;
ret_bit          : = GetMacAddr(ADR(mac_ab));
```

8.8.1.6 GetUserParam (only ETCxM)

Declaration

```
FUNCTION GetUserParam: LREAL
VAR_INPUT
    index_dw      : DWORD;          (* Index of MK_TECHNOLOGY_DATA_x *)
END_VAR
```

Description

This function allows for the values of the machine constants MK_TECHNOLOGY_DATA_1 to MK_TECHNOLOGY_DATA_4 to be read. The value *index_dw* to be passed corresponds to the position (starting with 0) numbered sequentially from first entry of MK_TECHNOLOGY_DATA_1 to the last entry of MK_TECHNOLOGY_DATA_4 .

Example

```
MK_TECHNOLOGY_D 1.1, 0, 0, 0, 0, 0, 0, 0, 0;
ATA_1
MK_TECHNOLOGY_D 0, 4.5, 3, -5.7, 0, 0, 0, 0, 0;
ATA_2
wert_d           : DINT;
wert_lr          : LREAL;
wert_lr          : = GetUserParam(9); (* returns the value -1.1 *)
wert_di          : = LREAL_TO_DINT(GetUserParam(12)); (* returns the value 3 *)
```

8.8.1.7 IO_SET

Declaration

```
FUNCTION IO_SET: BOOL
VAR_INPUT
    byte_w          : WORD;
    bit_w           : WORD;
END_VAR
```

Description

The function inverts the state of an output. In the output byte *byte_w* (value range 0 ... 31) the bit *bit_w* (value range 0 ... 7) will be inverted. The return value of the function is of no consequence.

Example

```
IO_SET(1,3);        inverts the output Q0.10
```

8.8.1.8 Load_Param

Declaration

```
FUNCTION LOAD_PARAM: DINT
VAR_INPUT
    dateiname_s     : STRING(15);       (* ”device:filename” *)
    pindex_di       : DINT;             (* P field index *)
    anzahl_di       : DINT;             (* number of the P field values *)
END_VAR
```

Description

LOAD_PARAM reads *anzahl_di* P field values from the RAM disk (device: rd), the FLASHPROM (device: sd) or a floppy disk (device: fd), which was previously saved with SAVE_PARAM in the file, and enters the value after the index *pindex_di* into the P field.

A return value other than 0 signals an error (📖 391).

Example

```
ret_di          : DINT;
ret_di          := LOAD_PARAM('rd:daten.dat', 1200, 30);
```

30 P field values from the file ” daten.dat” are read from the RAM disk and stored after index 1200 in the P field.

8.8.1.9 READ_PARAM_INT (only ETCxC)

Declaration

```
FUNCTION READ_PARAM_INT: INT
VAR_INPUT
    IDX_DI          : DINT;             (* Parameter index *)
END_VAR
```

Description

The function returns the value of the parameter *idx_di* (as data type INT) from the P field.

8.8.1.10 READ_PARAM_DINT (only ETCxC)

Declaration

```
FUNCTION READ_PARAM_DINT: DINT
VAR_INPUT
    IDX_DI          : DINT;          (* Parameter index *)
END_VAR
```

Description

The function returns the value of the parameter *idx_di* (as data type DINT) from the P field.

8.8.1.11 READ_PARAM_REAL (only ETCxC)

Declaration

```
FUNCTION
VAR_INPUT
    IDX_DI          : DINT;          (* Parameter index *)
END_VAR
```

Description

The function returns the value of the parameter *idx_di* (as data type REAL) from the P field.

8.8.1.12 READ_PARAM_LREAL (nur ETCxC)

Declaration

```
FUNCTION READ_PARAM_LREAL: LREAL
VAR_INPUT
    IDX_DI          : DINT;          (* Parameter index *)
END_VAR
```

Description

The function returns the value of the parameter *idx_di* (as data type LREAL) from the P field.

8.8.1.13 READ_SYSPARAM (only ETCxC)

Declaration

```
FUNCTION READ_SYSPARAM: WORD
VAR_INPUT
    IDX_I           : DINT;
END_VAR
```

Description

READ_SYSPARAM reads a value from the operating data.

Example

```
wert_w          : WORD;
wert_w          : = READ_SYSPARAM(101);
```

Returns the value of the operating data word 101.

8.8.1.14 READ_TOOLDATA (only ETCxC)

Declaration

```
FUNCTION READ_TOOLDATA: INT
VAR_INPUT
    tnr_dw          : DWORD;    * T (tool number) *)
    anzahl_dw       : DWORD;    (* number of tool data *)
    puffer_p        : DINT;     (* Address after which the data should be stored *)
END_VAR
```

Description

The function reads the tool data for the defined T (tools).

Example

```
TYPE TOOL_TR
STRUCT
    radius_lr           : LREAL; (* tool radius in mm *)
    typ_lr              : LREAL; (* tool type *)
    schwester_lr        : LREAL; (* no. of the sister tool *)
    original_lr         : LREAL; (* no. of the original tool *)
    magazin_lr          : LREAL; (* magazine location *)
    s_zugeordnet_lr     : LREAL; (* S allocated to the magazine location *)
    s_zeit_vorgabe_lr   : LREAL; (* default tool life *)
    s_zeit_aktuell_lr   : LREAL; (* actual tool life *)
    wz_aktuell_lr       : LREAL; (* actual tool *)
    drehzahl_lr         : LREAL; (* tool speed *)
END_STRUCT
END_TYPE
    wzdaten_tr          : TOOL_TR;
    ret_i               : INT;
    ret_i               := READ_TOOLDATA(2, 10, ADR(wzdaten_tr));
```

The tool data for tool 2 (T2) are stored in the structural variable *wzdaten_tr*.

8.8.1.15 SAVE_PARAM (only ETCxC)

Declaration

```
FUNCTION SAVE_PARAM: DINT
VAR_INPUT
    dateiname_s     : STRING(15);     (* ”device:filename” *)
    pindex_di       : DINT;           (* P field index *)
    anzahl_di       : DINT;           (* number of the P field values *)
END_VAR
```

Description

SAVE_PARAM saves *anzahl_di* P field values after index *pindex_di* under the file name ” dateiname_s” on the RAM disk (device: rd), in the FLASHPROM (device: sd) or on a floppy disk (device: fd).

A return value other than 0 signals an error (📖 391).

Example

```
ret_di          : DINT;
ret_di          : = SAVE_PARAM('rd:daten.dat', 1200, 30);
```

30 P field values are saved after index 1200 in the file ”daten.dat” on the RAM disk.

8.8.1.16 SETINPUT_BIT

Declaration

```
FUNCTION SETINPUT_BIT: BOOL
VAR_INPUT
    WORD_W          : WORD;           (* No. of the data word *)
    bit_w           : WORD;           (* input of the IO card *)
END_VAR
```

Description

The function sets a bit in the input process image. In the input*process image of the data word word_w (value range 0 ... 63) a bit bit_w* (value range 0 ... 31) is set. This allows for a test environment to be created (without hardware).

The return value FALSE indicates a wrong transfer parameter.

8.8.1.17 SETINPUT_WORD

Declaration

```
FUNCTION SETINPUT_WORD: BOOL
VAR_INPUT
    WORD_W      : WORD;      (* No. of the data word *)
    VAL_W       : WORD;
END_VAR
```

Description

The function executes a boolean OR operation with the value *val_w* and the data word *word_w* of the input process image and writes the result to the same data word.

Return value FALSE indicates a wrong transfer parameter.

The function allows the developer to create a test environment (without hardware).

8.8.1.18 SINGLEBLOCK (only ETCxC)

Declaration

```
FUNCTION SINGLEBLOCK: BOOL
VAR_INPUT
    Sentence string  : STRING(80);   (* sentence string *)
    PARAMETER_P      : DINT          (* Address of a structure *)
END_VAR
```

Description

This function sends an individual sentence to the NC computer. (The sentence string must be used as described under SPSERROR.)

Example 1

```
SINGLEBLOCK('G0 X100.2 Y50.73', 0);
```

An individual sentence without parameter is sent to the NC computer.

Example 2

```
TYPE PARAMETER_TR
STRUCT
    pos_X_lr        : LREAL;
    pos_Y_di        : DINT;
END_STRUCT
END_TYPE
param_tr:           PARAMETER_TR;
SINGLEBLOCK(>G0 X%f Y%d', ADR(param_tr));
```

An individual sentence with parameters is sent to the NC computer.

8.8.1.19 SPSERROR

Declaration

```
FUNCTION SPSERROR: BOOL
VAR_INPUT
    fehler_di]      : DINT;         (* error number *)
    klasse_dw       : DWORD;        (* error class *)
    formatstring    : STRING(80);   (* Format string *)
    PARAMETER_P     : DINT;         (* Address of a structure containing the parameters *)
END_VAR
```

Description

The function places an error message which is displayed on a connected control terminal. The error number *fehler_di* also serves as a reference for an error text. The error texts can be displayed on a connected PC (for both ETC variants) or on the ETC MMI (only ETCxC).

The error classes *klasse_dw* are divided as follows:

1: local minor error

2: local major error

3: global minor error

4: global major error

Error classes 3 and 4 cause the program to be cancelled or result in an axis movement.

Via the error texts references by the error numbers additional information can also be output. Via the format string additional text or numerical values can be transferred.

Format string: The % sign in a format string opens a format definition with the general form:

% Flag output field lengths Accuracy Data type definition

Flag	
<nothing>	flush right, leading spaces / nulls
-	flush left, following spaces / nulls
+	Always output operational sign

Output field length	
0n	min. n digits, fill with nulls
n	min. n digits, fill with spaces

Accuracy	
<nothing>	6 digits
.0	do not output a decimal point
.n	output max. n digits after the decimal point

Data type (CoDeSys)	Data type definition (format string)
DWORD	u
DINT	D
REAL	f e
LREAL	lf le
STRING	s

The return value of the function is of no consequence.

Example 1

```
SPSERROR(304, 4, 'Spindle does not start', 0);
```

Example 2

```
TYPE PARAMETER_TR
STRUCT
    jobno_dw        : DWORD;
    text_ps         : POINTER TO STRING;
    pos_lr          : LREAL;
END_STRUCT
END_TYPE
param_tr
string_s
param_tr.jobno_dw
param_tr.text_ps
param_tr.pos_lr
SPSERROR(303, 1, 'Auftragsnr%u %s %9.3lf', ADR(param_tr));
```

8.8.1.20 SETLANGUAGE (only ETCxM)

Declaration

```
FUNCTION SETLANGUAGE: BOOL
VAR_INPUT
    Language        : INT;
END_VAR
```

Description

This function switches between different output text languages. Language = 0 selectes the output text files ncrspch0.txt/spsspch0.txt, language = 1 the files ncrspch1.txt/spsspch1.txt etc.

The return value of the function is of no consequence.

Example

```
SETLANGUAGE(4);
```

Output text files ncrspch4.txt and spsspch4.txt are selected.

8.8.1.21 STRTOF

Declaration

```
FUNCTION STRTOF: REAL
VAR_INPUT
    SString         : STRING(255);
    PIndex          : POINTER TO INT;
END_VAR
```

Description

The function converts the characters in *sString* after position *Index* into a REAL. Index contains the read-in end position after execution.

8.8.1.22 STRTOL10

Declaration

```
FUNCTION STRTOL10: DINT
VAR_INPUT
    SString         : STRING(255);
    PIndex          : POINTER TO INT;
END_VAR
```

Description

The function converts the characters in *sString* after position *Index* into a DINT. Index contains the read-in end position after execution.

8.8.1.23 SYSERROR

Declaration

```
FUNCTION SYSERROR: BOOL
VAR_INPUT
    fehler_di       : DINT;     (* error number *)
    klasse_dw       : DWORD;    (* error class *)
    formatstring    : STRING(80); (* Format string *)
    PARAMETER_P     : DINT;     (* Address of a structure containing the parameters *)
END_VAR
```

Description

The function SYSERROR is used like the function SPSERROR.

This function can be used to display control-internal errors.

8.8.1.24 TRACE

Declaration

```
FUNCTION TRACE: BOOL
VAR_INPUT
    Idx             : DINT;     (* Trace index *)
    formatstring    STRING(80); (* Format string *)
    PARAMETER_P     : DINT;     (* Address of a structure *)
END_VAR
```

Description

The trace function can be very useful when debugging dynamic processes. Each trace can be purposefully enabled or disabled in the diagnostics monitor. For this a number to identify the trace is issued when the trace function is called. The trace numbers *idx*100 ... 200 are reserved for the PLC, all others are reserved for the NC computer.

The trace buffer in the NC computer holds 512 trace messages. If the trace messages arrive quicker in the buffer than they can be sent via the diagnostics interface (RS232), the buffer can overrun and the messages may be lost.

The return value of the function is of no consequence.

Example

```
TYPE PARAMETER_TR
STRUCT
    param1_r            : REAL;
    param2_di           : DINT;
END_STRUCT
END_TYPE
param_tr:               PARAMETER_TR;
parameter_tr.param1_r   : = 30.2
parameter_tr.param2_di  : = 20;
TRACE(100, 'REAL:%f DINT:%d', ADR(param_tr));
```

8.8.1.25 WordWrap

Declaration

```
FUNCTION WordWrap: INT
VAR_INPUT
    string_s        : STRING(255);      (* String, which is to be reformatted *)
    linelen_di      : DINT;             (* maximum line length *)
END_VAR
```

Description

The function automatically enters a line break into the string if the line length exceeds the value stated. The return value is the resulting total length of the string.

Ensure that the string variable declared is large enough to accommodate the resulting string!

8.8.1.26 WRITE_PARAM_INT (only ETCxC)

Declaration

```
FUNCTION WRITE_PARAM_INT: BOOL
VAR_INPUT
    IDX_DI          : DINT;             (* Parameter index *)
    VAL_I           : INT;              (* value *)
END_VAR
```

Description

The function writes the value *val_i* at the location *idx_di* into the P field.

The return value FALSE indicates an error.

8.8.1.27 WRITE_PARAM_DINT (only ETCxC)

Declaration

```
FUNCTION WRITE_PARAM_DINT: BOOL
VAR_INPUT
    IDX_DI          : DINT;             (* Parameter index *)
    VAL_DI          : DINT;             * value *)
END_VAR
```

Description

The function writes the value *val_i* at the location *idx_di* into the P field.

The return value FALSE indicates an error.

8.8.1.28 WRITE_PARAM_REAL (only ETCxC)

Declaration

```
FUNCTION WRITE_PARAM_REAL: BOOL
VAR_INPUT
    IDX_DI      : DINT;     (* Parameter index *)
    VAL_R       : REAL;     (* value *)
END_VAR
```

Description

The function writes the value *val_i* at the location *idx_di* into the P field. The return value FALSE indicates an error.

8.8.1.29 WRITE_PARAM_LREAL (only ETCxC)

Declaration

```
FUNCTION WRITE_PARAM_LREAL: BOOL
VAR_INPUT
    IDX_DI      : DINT;     (* Parameter index *)
    VAL_LR      : LREAL;    (* value *)
END_VAR
```

Description

The function writes the value *val_i* at the location *idx_di* into the P field.

The return value FALSE indicates an error.

8.8.1.30 WRITE_SYSPARAM (only ETCxC)

Declaration

```
FUNCTION WRITE_SYSPARAM: BOOL
VAR_INPUT
    IDX_I       : INT;      (* 0 ... 127 *)
    VAL_W       : WORD;
END_VAR
```

Description

WRITE_SYSPARAM writes the value *val_w* to the location *idx_i* in the operating data.

Return value 0 means wrong index.

Example

```
ret_bit     : BOOL;
et_bit      := WRITE_SYSPARAM(101, 100);
```

Writes the value 100 into the operating data word 101. This defines that the PROGRAM PLC_PRG2 (if available) will be called cyclically every 100 * 10 ms.

8.8.1.31 Return codes of the functions LOAD, SAVE, LOAD_PARAM, SAVE_PARAM

Return code	Description
-1	EEPROM faulty
-2	Checksum error in EEPROM
-3	EEPROM full
-4	File is already open, access denied
-5	Too many files opened simultaneously
-6	EEPROM file list is full
-7	File does not exist
-93	Checksum error in RAM disk
-94	Not enough memory for creating the file buffer
-95	RAM disk file table is full
-96	RAM disk is full
-110	Unknown device name
-111	V24 interface to the floppy disk drive is busy
-112	Floppy disk drive is not available
-113	Timeout during access to floppy disk drive
-114	Error during access to floppy disk drive
-115	Floppy disk drive reports transmission error
-116	Floppy disk drive reports V24 overrun
-117	Floppy disk is write protected
-118	No floppy disk inserted
-119	Read/write error on the floppy disk
-121	File name contains illegal characters

Specially for READ_PARAM, WRITE_PARAM

Return code	Description
1	P field index outside the permitted range

8.8.2 V24 functions

8.8.2.1 ALLOCV24

Declaration

```
FUNCTION AllocV24: DINT
VAR_INPUT
    unit_di     : DINT;     (* Interface: COM1 (X3), COM2 (X4) *)
    pri_di      : DINT;     (* Priority (-128 ... +127) *)
END_VAR
```

Description

A V24 interface must be allocated with AllocV24() before it can be used. It can then be initialised as required with InitV24().

The request is only met if the interface is available or the request priority is higher than the actual priority. An allocated V24 interface can be released again with FreeV24(); this restores the previous state prior to the allocation.

Possible interfaces are COM1 (*unit_di* = 0) and COM2 (*unit_di* = 1). For the priority, values between -128 (lowest) and +127 (highest) can be used, with a priority of 127 meaning that the interface cannot be ”appropriated”.

The function returns the address of the allocated V24 request structure. If it returns 0 the interface could not be allocated.

Example

```
request_p       : DINT;
request_p       : = ALLOCV24(0, 110);
```

The first V24 interface (X3 connector) is allocated with a priority of 110.

8.8.2.2 INITV24

Declaration

```
FUNCTION InitV24: DINT
VAR_INPUT
    req_pr      : DINT;     (* Address of the V24 request structure *)
    mode_dw     : DWORD;
    flags_dw    : DWORD;
END_VAR
```

Description

This function is used to set up the interface parameters of a V24 interface allocated with AllocV24.

► *mode_dw* results from the OR combination of mode bits (📖 391).

► *flags_dw* results from the OR combination of flag bits (📖 391).

A return value other than 0 signals an error (📖 391).

Example

```
ret_di      : DINT;
requestV24_p    : DINT;
requestV24_p    : = ALLOCV24(1, 127);
ret_di      : = InitV24(requestV24_p,
mode_no_parity OR mode_8_bits_per_char OR
mode_no_rts_cts_control OR mode_stop_bit_len_2 OR
mode_rx_baudrate_9600 OR mode_tx_baudrate_9600 OR
mode_rx_buffersize_2k OR mode_tx_buffersize_2k,flags_no_protocol);
```

8.8.2.3 FREEV24

Declaration

```
FUNCTION FreeV2: BOOL
VAR_INPUT
    req_pr      : DINT;     (* Address of the V24 request structure *)
END_VAR
```

Description

This function releases the access rights to the allocated V24 interface and returns the interface to its original state.

The return value of the function is of no consequence.

Example

```
requestV24_p    : DINT;
requestV24_p    : = ALLOCV24(1, 127);
REEV24(requestV24_p);
```

8.8.2.4 READBLOCKV24

Declaration

```
FUNCTION ReadBlockV24: DINT
VAR_INPUT
    pRequest    : DINT;    (* Address of the V24 request structure *)
    pBuffer     : DINT;    (* Address of a data buffer *)
    BufSize     : INT;     (* Buffer size in byte *)
END_VAR
```

Description

The function reads max. *BufSize* characters in the stated data buffer *pBuffer* from the V24 reception buffer.

The function returns the number of characters read from the reception buffer.

8.8.2.5 READV24

Declaration

```
FUNCTION ReadV24: DINT
VAR_INPUT
    req_pr      : DINT;    (* Address of the V24 request structure *)
END_VAR
```

Description

Reading a character from the reception buffer. The reception buffers is implemented as FIFO and is read and written to simultaneously.

This function always provides an immediate return. Return values smaller than 0 indicate an error (📖 391). If the FIFO is empty, -1 (EOF) will be returned.

Example

```
requestV24_p    : DINT;
zeichen_di      : DINT;
zeichen_di      : = READV24(requestV24_p);
```

8.8.2.6 WRITEBLOCKV24

Declaration

```
FUNCTION WriteBlockV24: DINT
VAR_INPUT
    pRequest    : DINT;    (* Address of the V24 request structure *)
    pBuffer     : DINT;    (* Address of a data buffer *)
    BufSize     : INT;     (* number of the characters to be written *)
END_VAR
```

Description

The function tries to insert *BufSize* characters of the stated address *pBuffer* into the V24 send buffer. If there is sufficient space in the send buffer all *BufSize* characters will be entered.

The function returns the number of the characters written into the send buffer.

8.8.2.7 WRITEV24

Declaration

```
FUNCTION WriteV24: DINT
VAR_INPUT
    req_pr      : DINT;     (* Address of the V24 request structure *)
    chr_di      : DINT;     (* character to be written *)
END_VAR
```

Description

Writing a character into the send buffer. The send buffer is implemented as FIFO and is read and written a character at a time.

This function always provides an immediate return even if the FIFO is full. In this case the character will not be written and -1 (EOF) will be returned. Generally a return value of less than 0 indicates an error (📖 391).

Example

```
requestV24_p        : DINT;
zeichen_di          : DINT;
ret_di              : DINT;

zeichen_di          : = 65;
ret_di              : = WRITEV24(requestV24_p, zeichen_di);
```

8.8.2.8 CLRRXBUFFER

Declaration

```
FUNCTION ClrRxBuffer: BOOL
VAR_INPUT
    req_pr      : DINT;     (* Address of the V24 request structure *)
END_VAR
```

Description

This function can be used to delete the reception FIFO completely.

The return value of the function is of no consequence.

Example

```
requestV24_p        : DINT;
requestV24_p        : = ALLOCV24(1, 127);
CLRRXBUFFER(requestV24_p);
```

8.8.2.9 CLRTXBUFFER

Declaration

```
FUNCTION ClrTxBuffer: BOOL
VAR_INPUT
    req_pr          : DINT;          (* Address of the V24 request structure *)
END_VAR
```

Description

This function can be used to delete the sender FIFO.

The return value of the function is of no consequence.

Example

```
requestV24_p        : DINT;
requestV24_p        : = ALLOCV24(1, 127);
CLRTXBUFFER(requestV24_p);
```

8.8.2.10 Global constants for V24 functions

(* error codes: *)	
V24_WRONG_UNIT_KDI	: DINT := -100; (* wrong V24 interface number *)
V24_OUTOFMEMORY_KDI	: DINT := -101; (* not enough RAM for creating the V24 FIFO buffer *)
V24_OVERRUN_ERROR_KDI	: DINT := -102; (* Overrun *)
V24_PARITY_ERROR_KDI	: DINT := -103; (* Parity error *)
V24_FRAMING_ERROR_KDI	: DINT := -104; (* Framing error *)
V24_RECEIVED_BREAK_KDI	: DINT := -105; (* Received break *)
(* Mode bits: *)	
MODE_EVEN_PARITY_KDW	: DWORD :=16#00000000;
MODE_ODD_PARITY_KDW	: DWORD := 16#00000004;
MODE_NO_PARITY_KDW	: DWORD := 16#00000010;
MODE_7_BITS_PER_CHAR_KDW	: DWORD :=16#00000000;
MODE_8_BITS_PER_CHAR_KDW	: DWORD := 16#00000001;
MODE_RTS_CTS_CONTROL_KDW	: DWORD := 16#00001080;
MODE_NO_RTS_CTS_CONTROL_KDW	: DWORD :=16#00000000;
MODE_STOP_BIT_LEN_1_KDW	: DWORD :=16#00000000;
MODE_STOP_BIT_LEN_2_KDW	: DWORD := 16#00000800;
MODE_BAUDRATE_2400_KDW	: DWORD := 16#00080000;
MODE_BAUDRATE_4800_KDW	: DWORD := 16#00090000;
MODE_BAUDRATE_9600_KDW	: DWORD := 16#000B0000;
MODE_BAUDRATE_19200_KDW	: DWORD := 16#000C0000;
MODE_BAUDRATE_38400_KDW	: DWORD := 16#000E0000;
MODE_BAUDRATE_57600_KDW	: DWORD := 16#000D0000;
MODE_BAUDRATE_115200_KDW	: DWORD := 16#000F0000;
MODE_RX_BUFFERSIZE_1K_KDW	: DWORD := 16#0A000000;
MODE_RX_BUFFERSIZE_2K_KDW	: DWORD := 16#0B000000;
MODE_RX_BUFFERSIZE_3K_KDW	: DWORD := 16#0C000000;
MODE_RX_BUFFERSIZE_4K_KDW	: DWORD := 16#0D000000;
MODE_TX_BUFFERSIZE_1K_KDW	: DWORD := 16#A0000000;
MODE_TX_BUFFERSIZE_2K_KDW	: DWORD := 16#B0000000;
MODE_TX_BUFFERSIZE_3K_KDW	: DWORD := 16#C0000000;
MODE_TX_BUFFERSIZE_4K_KDW	: DWORD := 16#D0000000;
(* Flag bits: *)	
FLAGS_NO_PROTOCOL_KDW	: DWORD :=16#00000000;
FLAGS_PLC_PROTOCOL_KDW	: DWORD := 16#00000001;
FLAGS_XON_XOFF_PROTOCOL_KDW	: DWORD := 16#00000002;
FLAGS_NAK_ON_RX_ERROR_KDW	: DWORD := 16#00000100;
FLAGS_MELD_RX_ERROR_KDW	: DWORD := 16#00000200;
FLAGS_USER_POLLING_KDW	: DWORD := 16#00001000;
FLAGS_CHANNEL_STOLEN_KDW	: DWORD := 16#80000000;

8.8.3 FILE IO functions

8.8.3.1 LOAD

Declaration

```
FUNCTION LOAD: DINT
VAR_INPUT
    Name          : STRING(15);   (* 'device:file name' *)
    daten_paten_p : DINT;         (* Address after which the data should be stored *)
    len           : INT;          (* Length of the data in bytes *)
END_VAR
```

Description

This function is used to read data from a device (📖 390).

From the file *name len* bytes will be read from the defined device and stored after the the memory address *daten_p*. Without device definition it will be attempted to read the file from the FLASHPROM.

A return value smaller than 0 signals an error (📖 391).

Example

```
werte_adi     : ARRAY[0..5] OF DINT;
anzahl_di     : DINT;
anzahl_di     := LOAD('RD:daten.dat', ADR(werte_adi), 24);
```

Six DINT values from the file "daten.dat" are read from the RAM disc and stored in the *Array werte_di* .

8.8.3.2 SAVE

Declaration

```
FUNCTION SAVE: DINT
VAR_INPUT
    Name          : STRING(15);   (* 'device:file name' *)
    daten_p       : DINT;         (* Address after which the data should be stored *)
    len           : INT;          (* Length of the data in bytes *)
END_VAR
```

Description

This function can be used to save data on a device (📖 390).

Under *name len* bytes are stored after memory address *daten_p* on the stated device. Without device definition the file will be written to the FLASHPROM.

A return value of less than 0 indicates an error while saving (📖 391).

Example

```
werte_adi     : ARRAY[0..5] OF DINT;
anzahl_di     : DINT;
anzahl_di     := SAVE('RD:daten.dat', ADR(werte_adi), 24);
```

Six DINT values from the array *werte_di* are saved in the file " daten.dat" on the RAM disk.

8.8.3.3 SetCurrentPath

Declaration

```
FUNCTION SetCurrentPath: DINT
VAR_INPUT
    path_s          : STRING(40);       (* device name and path *)
END_VAR
```

Description

Defines the actual default drive and the directory path for the file IO functions of the device drivers (📖 390). With the directory path the IP address of the CNC data server (network disk) can be defined.

Return value: The function returns an error code (📖 391).

Example

```
SetCurrentPath('ND:172.16.5.66');
```

8.8.3.4 SYSOPENFILE

Declaration

```
FUNCTION SYSOPENFILE : DINT
VAR_INPUT
    filename_s      : STRING(15);
    accessmode_di   : DINT;
END_VAR
```

Description

The function is used to open a file on the defined device (📖 390). It returns a file handle which is required for the functions SysReadFile() and SysWriteFile(). A total of 4 files can be opened simultaneously.

Parameters

filename_s:	Name of the file to be opened. Optionally with preceding device ID and colon.
accessmode_di:	Access mode O_RDONLY or O_WRONLY (defined in gobal constants)

Return value: file handle (greater or equal to 0 if successful, less than 0 in case of an error (📖 391).

Example

```
handle_di       : DINT;
handle_di       := SYSOPENFILE('RD:data.dat', O_RDONLY);
```

8.8.3.5 SYSCLOSEFILE

Declaration

```
FUNCTION SYSCLOSEFILE: DINT
VAR_INPUT
    handle_di       : DINT;
END_VAR
```

Description

The function is used to close a file previously opened with SysOpenFile(). The content of the FLASHPROMs is only updated in the FLASHPROM after the last file opened for write-access has been closed. The return value of the function is less than 0 for an error (all error codes are defined in the global constants of the library).

Parameters

handle_di	File handle (returned by SysOpenFile())

8.8.3.6 SYSREADFILE

Declaration

```
FUNCTION SYSREADFILE: DINT
VAR_INPUT
    handle_di       : DINT;    (* File handle returned by SysOpenFile *)
    buffer_p        : DINT;    (* Address of a buffer where the data are stored *)
    maxlen_di       : DINT;    (* Buffer length in byte *)
END_VAR
```

Description

The function is used for reading blocks (in sequence) from a file previously opened with SysOpenFile(). A maximum of *maxlen_di* characters are read from the file into the buffer defined with *buffer_p*. Less than *maxlen_di* characters will only be read if the file end is reached before or a read error occurs.

Return value: Actual number of read characters; 0 for file end or less than 0 for error (📖 391).

Example

```
handle_di       : DINT;
num_di          : DINT;
ret_di          : DINT;
buffer_from     : ARRAY[0..100] OF BYTE;

handle_di       := SYSOPENFILE('test.dat', O_RDONLY);
num_di          := SYSREADFILE(handle_di, ADR(buffer_from), 20);
ret_di          := SYSCLOSEFILE(handle_di);
```

8.8.3.7 SYSREADLINE

Declaration

```
FUNCTION SYSREADLINE: DINT
VAR_INPUT
    handle_di       : DINT;    (* File handle returned by SysOpenFile *)
    buffer_p        : DINT;    (* Address of a buffer where the data are stored *)
    maxlen_di       : DINT;    (* Buffer length in byte *)
END_VAR
```

Description

The function is used for the reading of lines (in sequence) from a file previously opened with SysOpenFile(). Reading takes place up to the line end but with a maximum of *maxlen_di* characters. The is only terminated early if less than *maxlen_d* characters were read. The line feed at the end of the line is also read.

Return value: Actual number of read characters; 0 for file end or less than 0 for error (📖 391).

8.8.3.8 SYSWRITEFILE

Declaration

```
FUNCTION SYSWRITEFILE: DINT
VAR_INPUT
    handle_di       : DINT;    (* File handle returned by SysOpenFile *)
    buffer_p        : DINT;    (* Address of a uffer *)
    len_di          : DINT;    (* number of the characters to be written *)
END_VAR
```

Description

The function is used to write blocks (in sequence) into a file previously opened with SysOpenFile(). *len_di* characters will be written to the file. If an error occurs during write, the file will be deleted and a value of less than 0 returend.

Return value: Number of data written, or less than 0 in case of error (📖 391).

Example

```
handle_di       : DINT;
num_di          : DINT;
ret_di          : DINT;
buffer_s        : STRING(10) := 'Hello';

handle_di       := SYSOPENFILE('test.dat', O_WRONLY);
num_di          := SYSWRITEFILE(handle_di, ADR(buffer_s), 5);
ret_di          := SYSCLOSEFILE(handle_di);
```

8.8.3.9 SYSREMOVEFILE

Declaration

```
FUNCTION SYSREMOVEFILE: DINT
VAR_INPUT
    filename_s      : STRING(15);       (* file name *)
END_VAR
```

Description

The function is used to delete a file from the FLASHPROM, the RAM disk or the FLOPPY.

Parameters

filename_pc:	Name of the file to be deleted. Optionally with preceding device ID and colon 🕮 390.

8.8.3.10 SYSDISKFORMAT

Declaration

```
FUNCTION SYSDISKFORMAT: DINT
VAR_INPUT
    device_s        : STRING(15);
END_VAR
```

Description

The function formats the FLASHPROM, the RAM disk or the FLOPPY. All data will be deleted!

Parameters

device_pc:	Device name 🕮 390. Empty string (" ") for the default device (FLASHPROM).

Return value: Less than 0 for an error (all errors are defined in the global constants of the library).

8.8.3.11 SYSFIRSTFILE, SYSNEXTFILE

Declaration

```
FUNCTION SYSFISRTFILE: DINT
VAR_INPUT
    info_p      : DINT;          (* Address of a variable of type FILEINFO_TR *)
    pattern_s   : STRING(15);    (* search pattern *)
END_VAR
```

```
FUNCTION SYSNEXTFILE: DINT
VAR_INPUT
    info_p      : DINT;          (* Address of a variable of type FILEINFO_TR *)
    pattern_s   : STRING(15);    (* search pattern *)
END_VAR
```

Description

These functions are required to read the file list of the FLASHPROM, the RAM disk or the FLOPPY. With SysFirstFile() the stated file info structure is intialised and the information for the first file stored in structure. All subsequent calls of SysNextFile(), with the same file info structure, store the information of the next file in sequence in the structure.

The file info structure is structured as follows:

```
TYPE FILEINFO_TR
STRUCT
    filename_s      : STRING(15); (* file name *)
    filesize_dw     : DWORD; (* file size in byte *)
    filenum_di      : DINT; (* No. of the directory entry *)
END_STRUCT
END_TYPE
```

Parameters

info_p:	Pointer to the file info structure where the file name and the file length are to be stored.
pattern_s:	Search pattern with MS-DOS wild cards. Each time the next file whose file name matches the pattern will be returned.

Return value: 0, if OK.. DSK_FILE_NOT_EXIST_KDI, if no further files available. All other values indicate an error (all errors are defined in the global constants of the library).

Example

```
info_st     : FILEINFO_TR;
err_di       DINT;

err_di = SYSFIRSTFILE(ADR(info_st), 'SD:*.DIN'); (* fetch info on the first file *)
err_di = SYSNEXTFILE(ADR(info_st), 'SD:*.DIN'); (* fetch info on the next file *)
```

8.8.3.12 SYSDISKINFO

Declaration

```
FUNCTION SYSDISKINFO: DINT
VAR_INPUT
    info_p      : DINT;      (* Address of a variable of type DISKINFO_TR *)
    pattern_s   : STRING(15); (* search pattern *)
END_VAR
```

Description

The function detects the usage data for a device and stores it in the disk info structure defined.

The disk info structure is structured as follows:

```
TYPE DISKINFO_TR
STRUCT
    blocksize_w   : WORD; (* Size of a logical block in bytes *)
    blocknumb_w   : WORD; (* Formattet total capacity in blocks *)
    blockused_w   : WORD; (* Used capacity in blocks *)
    blockfree_w   : WORD; (* Free capacity in blocks *)
END_STRUCT
END_TYPE
```

Parameters

info_p:	Pointer to the disk info structure.
device_pc:	Pointer to the device name 📖 390. Empty string (" ") for the default device.

8.8.3.13 Device driver

The control supports different devices with write (w), and for some devices, read (r) access. The access takes place via a file system. A file must be opened using the function SysOpenFile() before it can be accessed. With the exception of the device ”Printer” several files can be opened simultaneously on a device. In total four simultaneously opened files are possible. The file names must comply with the DOS 8.3 convention, i.e. must contain a maximum of 8 characters for the file name and 3 characters for the file extension (no special characters must be used).

Device	ID	Access	Description
Silicone disk	SD	rw	This driver allows access to the Flash PROM of the control.
RAM Disk	RD	rw	This driver allows access to the so-called RAM Disk (battery buffered RAM).
Net Disk	ND	rw	This driver allows access to the hard drive of a PC. The device can only be used if the control has an Ethernet interface. On the PC the Lenze ETC Data Server must have been started. The PLC can access the directory which has been entered into the configuration file of the server. To create the connection between the control and the PC the control given the IP address of the PC via the function SetCurrentPath(). For this it is necessary to enter a fixed IP address in the network settings of the PC.
PC Disk	PC	rw	This driver allows access to the hard drive of a PC. The device can only be used if an application is running on the PC which uses the Lenze Gateway (e.g. ETC MMI). Using the Gateway configuration software a directory can be defined which the PLC can access.
Program memory (only ETCxC)	PS	W	This driver allows DIN programs to write tot he internal DIN program memory via the PLC. It must be ensured that the file name must also be identical to the program number, therefore the DIN program must not contain a program number (with % command). Example: SysOpenFile('PS:12.din', O_WRONLY); opens a DIN program with the program number %12.

8.8.3.14 Global constants for File IO functions

Constant	Value	Meaning
DSK_EEPROM_NOT_READY_KI	-1	Flash PROM faulty
DSK_EEPROM_CHECKSUM_KI	-2	Checksum error in flash disk
DSK_EEPROM_DISK_FULL_KI	-3	Flash disk is full, file cannot be accepted
DSK_ACCESS_DENIED_KI	-4	File is already open, access denied
DSK_TOO_MANY_FILES_KI	-5	Too many files opened simultaneously, access denied
DSK_EEPROM_DIR_FULL_KI	-6	Flash disk file directory is full, too many files
DSK_FILE_NOT_EXIST_KI	-7	File does not exist
DSK_INVALID_ACCESSMODE_KI	-8	Illegal access mode for SysOpenFile
DSK_UNKNOWN_EEPROM_TYPE_KI	-9	Unknown flash type encountered
DSK_FILE_NOT_OPEN_KI	-10	File not open for read or write access
DSK_EEPROM_OUT_OF_MEMORY	-11	Flash disk: Not enough memory for setting up the flash disk
DSK_SRAM_TIMEOUT_KI	-91	RAM disk: XT3 expansion card does not respond
DSK_SRAM_COM_ERROR_KI	-92	RAM disk: Communnication error with XT3 expansion card
DSK_SRAM_CHECKSUM_KI	-93	Checksum error in RAM disk
DSK_SRAM_OUTOFMEMORY_KI	-94	RAM disk: not enough heap memory for creating the file buffer
DSK_SRAM_DIR_FULL_KI	-95	No more space in RAM disk directory, too many files
DSK_SRAM_DISK_FULL_KI	-96	RAM disk is full
DSK_SRAM_VALIDATE_ERR_KI	-97	RAM disk validation: File could not be recovered and was deleted
DSK_SRAM_BATT_EMPTY_KI	-98	RAM disk: Battery is empty, please replace
DSK_SRAM_RAM_ERROR_KI	-99	RAM disk: RAM error in CMOS RAM
DSK_UNKNOWN_DEVICE_KI	-110	Unknown device name
DSK_FLOPPY_V24_INUSE_KI	-111	V24 interface to the floppy disk drive already in use
DSK_FLOPPY_INUSE_KI	-112	Floppy disk drive not empty, access denied
DSK_FLOPPY_TIMEOUT_KI	-113	Timeout during access to floppy disk drive
DSK_FLOPPY_PROG_ERROR_KI	-114	Error during access to floppy disk drive
DSK_FLOPPY_TRANSMIT_ERROR_KI	-115	Floppy disk drive reports transmission error
DSK_FLOPPY_OVERRUN_ERROR_KI	-116	Floppy disk drive reports V24 overrun
DSK_FLOPPY_WRITE_PROTECT_KI	-117	Floppy disk is write protected
DSK_FLOPPY_NO_DISK_KI	-118	No floppy disk inserted
DSK_FLOPPY_NOT_FORMATTED_KI	-119	Read/write error on the floppy disk
DSK_FUNCTION_NOT_SUPPORTED_KI	-120	The function to be executed is not supported by the device driver
DSK_INVALID_FILENAME_KI	-121	File name contains illegal characters
DSK_PRINTER_v24_INUSE_KI	-122	V24 interface to the printer already in use
DSK_PRINTER_INUSE_KI	-123	Printer is already in use, access denied
DSK_PRINTER_TIMEOUT_KI	-124	Printer not connected or not ready
DSK_BLOCK_OUT_OF_RANGE	-200	File block number outside the permitted range
DSK_DUPLICATE_BLOCK_KI	-201	File block used twice
DSK_PCDISK_IOERR	-301	PC disk: Unknown IO error
DSK_DLL_TIMEOUT	-302	PC disk: Timeout during IO operation

Constant	Value	Meaning
DSK_DLL_OUT_OF_MEMORY	-303	PC disk: Insufficient memory during IO operation
DSK_NET_NOT_CONFIGURED	-331	Network disk: drive not initialised
DSK_NET_COM_ERR	-332	Network disk: Communication error, wrong command code
DSK_NET_ANSWER_ERR	-333	Network disk: Communication error, wrong acknowledgement
DSK_NET_TIMEOUT_ERR	-334	Network disk: Timeout during access to the drive
DSK_NET_CANNOT_OPFEN_FILE	-335	Network disk: Server reports error during FileOpen
DSK_NET_CANNOT_CLOSE_FILE	-336	Network disk: Server reports error during FileClose
DSK_NET_HANDLE_ERROR	-337	Network disk: Server reports handle error
DSK_NET_READ_ERROR	-338	Network disk: Server reports error during FileRead
DSK_NET_WRITE_ERROR	-339	Network disk: Server reports error during FileWrite
DSK_NET_FILE_NOT_EXISTS	-340	Netword disk: File does not exist
DSK_NET_DIR_NOT_CREATED	-341	Network disk: Server reports error during CreateDir
DSK_NET_DIR_NOT_DELETED	-342	Network disk: Server reports error during DeleteDir
DSK_NET_FILE_NOT_CREATED	-343	Network disk: Server reports error during CreateFile
DSK_NET_FILE_NOT_DELETED	-344	Network disk: Server reports error during DeleteFile
DSK_NET_BUSY	-345	Network disk: Drive busy, access denied
DSK_CMS_INIT_KI	-401	CMOS RAM invalid: Remaining data were deleted
DSK_CMS_BATT_EMPTY_KI	-402	CMOS RAM: Battery empty, please replace
DSK_CMS_OLD_BOOTLOADER_KI	-403	CMOS RAM: old bootloader version, please update
O_RDONLY	0	
O_WRONLY	1	

8.8.4 Memory access functions

8.8.4.1 DEFDATATYPES

Declaration

```
FUNCTION DEFDATATYPES: BYTE
VAR_INPUT
    iRange          : INT;
    sDescriptor     : STRING(255);
END_VAR
```

Description

The function defines the data types in DB2 and thereby the required byte swapping during the data transfer between NCR and MMI.

Currently, two ranges are being differentiated:

Range 1 of word 0 ... 127 contains data from the PLC to MMI, range 2 of word 128 ... 191 contains data from MMI to the PLC. A descriptor string is used to define the data types. The descriptor string has the following format:

<number><type><number><type>...

Number is a decimal number and defines then data quantity for the following type.

Type is a lower-case letter describing the size of the data type as follows:

Type	Codesys data type	Size
B	BYTE	8 Bit
W	BOOL, WORD, INT	16 Bit
D	DWORD, DINT, REAL	32 Bit
l	LREAL	64 Bit

The function returns a value of 1 if the function has been completed successfully, otherwise a value of 0.

Example

At the beginning of DB2 in range 1 of *%MW2.0* to *%MW2.16* there are 10 BYTE, 2 WORD, 3 DINT and 1 LREAL. In range 2 of *%MW2.128* to *%MW2.158* there are 5 WORD, 8 DWORD and 5 WORD in the order stated. The definition is given as follows:

```
DEFDATATYPES(1,'10b2w3d1l')
DEFDATATYPES(2,'5w8d10w')
```

If a message interface in DB2 is used between PLC and HMI and the messages use different data types than WORD, additional adaptations are required. For this purpose the ETC has two type-related functions:

GET_BYTE	PUT_BYTE
GET_WORD	PUT_WORD
GET_INT	PUT_INT
GET_DWORD	PUT_DWORD
GET_DINT	PUT_DINT
GET_REAL	PUT_REAL
GET_LREAL	PUT_LREAL

The functions GET_... read the corresponding data type from the address stated and carry out the necessary byte swapping for the HMI. The functions PUT_... write the corresponding data type swapped to the address stated.

Condition for the data to arrive correctly at the HMI when using the above functions is the definition of the user data of the messsage as byte via DEFDATATYPES. For the standard HMI this looks as follows:

```
DEFDATATYPES(1,'96w4w52b2w'); (* 96 WORD, 4 WORD, 52 BYTE, 2 WORD *)
DEFDATATYPES(2,'32w4w52b2w'); (* 32 WORD, 4 WORD, 52 BYTE, 2 WORD *)
```

The definition results from the definition of the data in DB2 which are transferred between ETC-MMI and the PLC:

Data word	Name	Direction
000.00 – 015.15	States (256 Bits)	PLC → HMI
016.00 – 079.15	Displays (64 data word)	PLC → HMI
080.00 – 095.15	Notices/static errors (256 Bits)	PLC → HMI
096.00 – 096.15	Message buffer: acknowledgment counter	PLC → HMI
097.00 – 097.15	Message buffer: counter 1	PLC → HMI
098.00 – 125.15	Message buffer: User data (28 data words)	PLC → HMI
126.00 – 126.15	Message buffer: counter 2	PLC → HMI
127.00 – 127.15	PLC monitoring running	PLC → HMI
128.00 – 143.15	States (256 Bits)	PLC ← HMI
144.00 – 159.15	Keys (256 Bits)	PLC ← HMI
160.00 – 160.15	Message buffer: receipt counter	PLC ← HMI
161.00 – 161.15	Message buffer: counter 1	PLC ← HMI
162.00 – 189.15	Message buffer: User data (28 data word)	PLC ← HMI
190.00 – 190.15	Message buffer: counter 2	PLC ← HMI
191.00 – 191.15	ETC-MMI monitoring running	PLC ← HMI

8.8.4.2 GET_BYTE, GET_WORD, GET_DWORD, GET_INT, GET_DINT, GET_REAL, GET_LREAL

Declaration

```
FUNCTION GET_TYPE : BYTE
VAR_INPUT
    pAddress        : DINT;          (* memory address *)
END_VAR
```

Description

The functions GET_TYPE read the corresponding data type TYPE from the address stated and carry out the required byte swapping (see also DEFDATATYPES).

8.8.4.3 MEMCOMP

Declaration

```
FUNCTION MEMCOMP: INT
VAR_INPUT
    pMem1           : DINT;          (* memory address *)
    pMem2           : DINT;          (* memory address *)
    diSize          : DINT;          (* number of bytes *)
END_VAR
```

Description

The memory areas Mem1 and Mem2 are compared in bytes.

Return value

0: if the content is identical

1: if the content of Mem1 > Mem2

-1: if the content of Mem1 < Mem2

8.8.4.4 MEMCOPY

Declaration

```
FUNCTION MEMCOPY: BOOL
VAR_INPUT
    ziel_p          : DINT;          (* memory address *)
    daten.dat       : DINT;          (* memory address *)
    anzahl_di       : DINT;          (* number of bytes *)
END_VAR
```

Description

The *anzahl_di* bytes after memory address *quelle_p* are read and entered after memory address *ziel_p*.

The return value of the function is of no consequence.

Example

```
ret_bit             : BOOL;
feld1_ab            : ARRAY[0..10] OF BYTE;
feld2_ab            : ARRAY[0..5] OF BYTE;

MEMCOPY(ADR(feld1_ab[4]), ADR(feld2_ab[2]), 4);
MEMCOPY(ADR(feld2_ab[0]), ADR(feld1_ab[2]), SIZEOF(feld2_ab));
```

8.8.4.5 MEMSET

Declaration

```
FUNCTION MEMSET: BOOL
VAR_INPUT
    pMem        : DINT;     (* memory address *)
    bValue      : DINT;     (* value *)
    dwSize      : DINT;     (* number of bytes *)
END_VAR
```

Description

dwSize bytes after memory address *pMem* with a value *bValue* will be written.

Example

```
feld_ab             : ARRAY[0..10] OF BYTE;
MEMSET( ADR(feld_ab[0]), 0, SIZEOF(feld_ab) );
```

8.8.4.6 OVESWAPPED

Declaration

```
FUNCTION MOVESWAPPED: DINT
VAR_INPUT
    pDestination    : DINT;         (* memory address *)
    pSource         : DINT;         (* memory address *)
    sDescstring     : STRING(255);  (* Data description string *)
END_VAR
```

Description

The function copies data from the memory address *pDestination* to the memory address *pSource*. The data are copied swapped in accordance with the descriptor string defined.

The descriptor string has the following format:

<number><type><number><type>...

Number is a decimal number and defines then data quantity for the following type.

Type is a lower-case letter describing the size of the data type as follows:

Type	Codesys data type	Size
B	BYTE	8 Bit
W	BOOL, WORD, INT	16 Bit
D	DWORD, DINT, REAL	32 Bit
l	LREAL	64 Bit

The function returns as value the number of bytes copied.

Example

```
TYPE DATA_TR :
STRUCT
    var1            : DINT;
    var2            : DINT;
END_STRUCT
END_TYPE

byte_di             : DINT;
descstring_s        : STRING(10) := '2d';
data_st             : DATA_TR := (var1:=16#FF, var2:=16#123456);
buf_ab              : ARRAY[0..20] OF BYTE;

byte_di := MOVESWAPPED(ADR(buf_from), ADR(data_st), descstring_s);
```

Result:

```
byte_di = 8, buf_from[0] = 16#FF, buf_from[4] = 16#56, buf_from[5] = 34, buf_from[6] :=
16#12
```

All other elements of the array are equal to 0.

8.8.4.7 PUT_BYTE, PUT_WORD, PUT_DWORD, PUT_INT, PUT_DINT, PUT_REAL, PUT_LREAL

Declaration

```
FUNCTION PUT_TYPE: BYTE
VAR_INPUT
    pAddress        : DINT;         * memory address *)
    Value           : TYPE;         (* value to be written *)
END_VAR
```

Description

The functions PUT_TYPE write the corresponding data type TYPE swapped to the address stated (see also DEFDATATYPES).

8.8.5 CANopen functions

8.8.5.1 CopDefineDS403

Declaration

```
FUNCTION CopDefineDS403: BOOL
VAR_INPUT                (* Defines the global database for a CanOpen control element in
                            accordance with DS403 *)
    NodeID               : BYTE;
    DataBase             : DINT;
    DataSize             : WORD;
    Datatypes            : STRING(255);
    ReadDataObject       : WORD;
    WriteDataObject      : WORD;
END_VAR
```

Parameters

NodeID	Node number of the control element
DataBase	Address of the global database with the input/output data of the control element
DataSize	Size of the global database in Byte, max. 1020 Byte
Datatypes	Descriptor string for defining the data types in the global database
ReadDataObject	Object number for reading the ”Local database”
WriteDataObject	Object number for describing the ”Local database”

Description

This function supplies the runtime system with a data range of the PLC for the data exchange with a CANopen control element. This data range is best created in the PLC as a data structure or an array and initialised with the data to be displayed. After calling this function this global database is copied once completely into the local database of the corresponding node. Afterwards only the modifications of the display data will be transferred by the control system.

ReadDataObject and WriteDataObject are the object numbers of the object directory of the CANopen display which must be used by the runtime system to read from and write to the local database.

The descriptor string has the following format:

<number><type><number><type>...

Number is a decimal number and defines the data quantity for the following type.

Type is a lower-case letter describing the size of the data type as follows:

Type	Codesys data type	Size
B	BYTE	8 Bit
W	BOOL, WORD, INT	16 Bit
D	DWORD, DINT, REAL	32 Bit

The return value is TRUE if the function has been executed successfully. Otherwise the function could not be executed, e.g. due to lack of memory.

Note!

The database should only be as large as required by the data actually used.

The PLC should always only update the data in the global database which are actually being displayed to prevent an unnecessary increase of the bus load on the CAN Bus. Cyclical changes during each PLC cycle should generally be avoided.

Example

```
TYPE DATABASE_TR :
STRUCT
    dummy_dw        : DWORD;
    pad1_dw         : DWORD;
    pad2_b          : BYTE;
    pad3_b          : BYTE;
    pad4_w          : WORD;
END_STRUCT
END_TYPE
data_st             : DATABASE_TR;
COPDEFINEDS403( 127, ADR(data_st), SIZEOF(data_st), '2d2b1w', 16#6700, 16#8100 );
```

The function defines a global database with the elements defined by the descriptor string for a CANopen control element with the node number 127, which has been inserted accordingly into the control configuration of CoDeSys.

8.8.5.2 CopXDefineDS403 (only ETCxM)

Declaration

```
FUNCTION CopXDefineDS403: BOOL          (* Defines the global database for a CANopen
                                           control element in accordance with DS403 for
                                           1/2 CAN Bus *)
VAR_INPUT
    CanNum_b        : BYTE
    NodeID_b        : BYTE
    DataBase_p      : DINT;
    Len_w           : WORD;
    Datatypes_s     : STRING(255);
    ReadDataObject  : WORD;
    WriteDataObject : WORD;
END_VAR
```

Parameters

CanNum_b:	Number of the CAN Bus (1 or 2)
NodeID_b:	Node number of the control element
DataBase_p:	Address of the global database with the input/output data of the control element
Len_w:	Size of the global database in Byte, max. 1020 Byte
Datatypes_s:	Descriptor string for defining the data types in the global database
ReadDataObject:	Object number for reading the ”Local database”
WriteDataObject:	Object number for describing the ”Local database”

Description

The functionality of this function is identical to the function CopDefineDS403(). It contains an additional parameter *CanNum_b*, which defines the number of the respective CAN Bus.

8.8.5.3 CopGetNodeState (only ETCxM)

Declaration

```
FUNCTION CopGetNodeState: BOOL
VAR_INPUT
    NodeID_b        : BYTE;             * CANopen ID of the module *)
END_VAR
```

Parameters

NodeID_b:	Node number of the CANopen node

Description

The return value TRUE signals that the corresponding CANopen module is present at the 1st CAN Bus and in an ”operational” state. Otherwise FALSE will be returned. This detection only works if node guarding has been enabled for the corresponding CANopen device.

8.8.5.4 CopReadObjekt (ETCxC and ETCxM at CAN1), DrvReadObject (ETCxC at CAN2)

Declaration

```
FUNCTION CopReadObject: BOOL            (ETCxC and ETCxM at CAN1)
(* Read-out of objects in the object directory of CANopen devices at CAN1 *)
VAR_INPUT
    NodeID      : BYTE;
    ObjectNum   : WORD;
    SubIndex    : BYTE;
    DataType    : WORD;
    Buffer      : DINT;
    BufSize     : WORD;
    Status      : POINTER TO BYTE;
END_VAR
```

```
FUNCTION DrvReadObject: BOOL            (ETCxC at CAN2)
(* Read-out of objects in the object directory of CANopen devices at CAN2 *)
VAR_INPUT
    NodeID      : BYTE;
    ObjectNum   : WORD;
    SubIndex    : BYTE;
    DataType    : WORD;
    Buffer      : DINT;
    BufSize     : WORD;
    Status      : POINTER TO BYTE;
END_VAR
```

Parameters

NodeID	Node number of the CANopen node
ObjectNum	Number of the object to be read
SubIndex	Index of the subobject to be read
DataType	Data type according to CiA DS301
Buffer	Address of the buffer in which the read data are stored
BufSize	Size of the buffer in Byte
Status	Address of a variable for filing the transfer status

Description

This function allows the PLC to read individual objects of CANopen devices. The reading process takes place in the background whilst the PLC continues working. The transfer queue can buffer up to 15 read/write requests.

By requesting the transfer status the PLC can check whether the transfer is still ongoing, has finished or whether an error has occurred during transfer. If several transfer requests are to be carried out in parallel, a separate status variable needs to be defined for each request.

The return value is TRUE if the read request could be passed to the transfer queue, otherwise the transfer queue is already full.

Status	Meaning
0	Inactive
1	Request in transfer queue
2	Transfer active
3	Transfer completed successfully
4	Transfer cancelled
5	Timeout

Example

```
status_b
devicetype_dw

CopReadObject(1, 16#1000, 0, COP_UNSIGNED32_KW, ADR(devicetype_dw),
SIZEOF(devicetype_dw), ADR(status_b));
```

8.8.5.5 CopXReadObject (ETCxM at CAN2)

Declaration

```
FUNCTION CopXReadObject: BOOL          (* Reading of objects in the object directory of
                                          CANopen devices at CAN2 *)
VAR_INPUT
    CanNum_b      : BYTE;
    NodeID_b      : BYTE;
    ObjectNum_w   : WORD;
    SubIndex_b    : BYTE;
    DataType_w    : WORD;
    Buffer_p      : DINT;
    BufSize_w     : WORD;
    Status_pb     : POINTER TO BYTE;
END_VAR
```

Parameters

CanNum_b:	Number of the CAN Bus (1 or 2)
NodeID_b:	Node number of the CANopen node
ObjectNum_w:	Number of the object to be read
SubIndex_b:	Index of the subobject to be read
DataType_w:	Data type according to CiA DS301
Buffer_p:	Address of the buffer in which the read data are stored
BufSize_w:	Size of the buffer in Byte
Status_pb:	Address of a variable for filing the transfer status

Description

The functionality of this function is completely identical to the function CopReadObjekt(). It contains an additional parameter *CanNum_b*, which defines the number of the respective CAN Bus.

8.8.5.6 CopWriteObject (ETCxC and ETCxM at CAN1), DrvWriteObject (ETCxC at CAN2)

Declaration

```
FUNCTION CopWriteObject: BOOL          (ETCxC and ETCxM at CAN1)
(* Description of objects in the object directory of CANopen devices at CAN1 *)
VAR_INPUT
    NodeID      : BYTE;
    ObjectNum   : WORD;
    SubIndex    : BYTE;
    DataType    : WORD;
    Buffer      : DINT;
    BufSize     : WORD;
    Status      : POINTER TO BYTE;
END_VAR
```

```
FUNCTION DrvWriteObject: BOOL          (ETCxC at CAN2)
(* Description of objects in the object directory of CANopen devices at CAN2 *)
VAR_INPUT
    NodeID      : BYTE;
    ObjectNum   : WORD;
    SubIndex    : BYTE;
    DataType    : WORD;
    Buffer      : DINT;
    BufSize     : WORD;
    Status      : POINTER TO BYTE;
END_VAR
```

Parameters

NodeID	Node number of the CANopen device
ObjectNum	Number of the object to be written
SubIndex	Index of the subobject to be written
DataType	Data type according to CiA DS301
Buffer	Address of the buffer in which the data to be written are stored
BufSize	Number of bytes to be written
Status	Address of a variable for filing the transfer status

Description

This function allows the PLC to write to individual objects of CANopen devices. The writing process takes place in the background whilst the PLC continues working. The transfer queue can buffer up to 15 read/write requests.

By requesting the transfer status the PLC can check whether the transfer is still ongoing, has finished or whether an error has occurred during transfer. If several transfer requests are to be carried out in parallel, a separate status variable needs to be defined for each request.

The function returns TRUE if the write request has been passed to the transfer queue, otherwise the transfer queue is already full.

Status	Meaning
0	Inactive
1	Request in transfer queue
2	Transfer active
3	Transfer completed successfully
4	Transfer cancelled
5	Timeout

Example

```
status_b        : BYTE;
wert_di         : DINT := 1000;

CopWriteObject(5, 16#607A, 0, COP_INTEGER32_KW, ADR(value_di), SIZEOF(value_di),
ADR(status_b));
```

Entering the target position for a DS402 drive.

8.8.5.7 CopXWriteObject (ETCxM at CAN2)

Declaration

```
FUNCTION CopXWriteObject: BOOL          (* Description of objects in the object directory
                                           of CANopen devices at CAN2 *)
VAR_INPUT
    CanNum_b        : BYTE;
    NodeID_b        : BYTE;
    ObjectNum_w     : WORD;
    SubIndex_b      : BYTE;
    DataType_w      : WORD;
    Buffer_p        : DINT;
    BufSize_w       : WORD;
    Status_pb       : POINTER TO BYTE;
END_VAR
```

Parameters

CanNum_b:	Number of the CAN Bus (1 or 2)
NodeID_b:	Node number of the CANopen device
ObjectNum_w:	Number of the object to be written
SubIndex_b:	Index of the subobject to be written
DataType_w:	Data type according to CiA DS301
Buffer_p:	Address of the buffer in which the data to be written are stored
BufSize_w:	Number of bytes to be written
Status_pb:	Address of a variable for filing the transfer status

Description

The functionality of this function is identical to the function CopWriteObject. It contains an additional parameter *CanNum_b*, which defines the number of the respective CAN Bus.

8.8.5.8 CopXSetOperational (only ETCxM)

Declaration

```
FUNCTION CopXSetOperational: BOOL      (* Setting the status operational for the
                                          CANopen node *)
VAR_INPUT
    CanNum_b    : BYTE;
    NodeID_b    : BYTE;
    Status_pb   : POINTER TO BYTE;
END_VAR
```

Parameters

CanNum_b:	Number of the CAN Bus (1 or 2)
NodeID_b:	Node number of the CANopen device
Status_pb:	Address of a variable for filing the transfer status

Description

The function enables the PLC to set the status of individual CANopen nodes to operational. A condition for this is that the node has been entered into the CoDeSys control configuration. The telegrams will be sent in the background whilst the PLC continues working.

By requesting the transfer status the PLC can check whether the message has been sent or an error has occurred. If several transfer requests are to be carried out in parallel, a separate status variable needs to be defined for each request.

The function returns TRUE if the write request has been passed to the transfer queue. Otherwise the transfer queue is already full.

Status	Meaning
0	Inactive
1	Request in transfer queue
2	Transfer active
3	Transfer completed successfully
4	Transfer cancelled

Example

```
status_b        BYTE;
wert_di         : DINT := 1000;

CopXSetOperational(1,5, ADR(status_b));
```

The node with address 5 at the 1st CAN Bus has its status set to operational.

8.8.5.9 CopXSetPreoperational (only ETCxM)

Declaration

```
FUNCTION CopXSetPreoparational: BOOL    (* Setting the status pre-operational for the
                                           CANopen node *)
VAR_INPUT
    CanNum_b    : BYTE;
    NodeID_b    : BYTE;
    Status_pb   : POINTER TO BYTE;
END_VAR
```

Parameters

CanNum_b:	Number of the CAN Bus (1 or 2)
NodeID_b:	Node number of the CANopen device
Status_pb:	Address of a variable for filing the transfer status

Description

This function enables the PLC to set the status of individual CANopen nodes to pre-operational. A condition for this is that the node has been entered into the CoDeSys control configuration. The telegrams will be sent in the background whilst the PLC continues working.

By requesting the transfer status the PLC can check whether the message has been sent or an error has occurred. If several transfer requests are to be carried out in parallel, a separate status variable needs to be defined for each request.

The function returns TRUE if the write request has been passed to the transfer queue. Otherwise the transfer queue is already full.

Status	Meaning
0	Inactive
1	Request in transfer queue
2	Transfer active
3	Transfer completed successfully
4	Transfer cancelled

Example

```
status_b        : BYTE;
wert_di         : DINT := 1000;

CopXSetPreoperational(1,5, ADR(status_b));
```

The node with address 5 at the 1st CAN Bus has its status set to pre-operational.

8.8.5.10 CopXEnableSync (only ETCxM)

Declaration

```
FUNCTION CopXEnableSync: DINT          (* Releasing the Sync message *)
VAR_INPUT
    CanNum_uc       : BYTE;
END_VAR
```

Parameters

CanNum_uc	Number of the CAN Bus (1 or 2)

Description

Calling this function enables the sending of the CANopen Sync message by the PLC. The Sync messages will be sent with the cycle time of the PLC task calling the function.

The function returns the following values:

Return value	Meaning
0	Function completed successfully
-1	CAN task is not ready
-3	Wrong CAN number
-4	The function has already been released
-5	The send buffer of the CAN Controller is already assigned

8.8.5.11 CopXDisableSync (only ETCxM)

Declaration

```
FUNCTION CopXDisableSync : DINT        (* Blocking the sending of Sync messages *)
VAR_INPUT
    CanNum_uc       : BYTE;
END_VAR
```

Parameters

CanNum_uc	Number of the CAN Bus (1 or 2)

Description

Calling this function disables the sending of the CANopen Sync message by the PLC. The blocking of the Sync messages can only take place before the PLC task which enabled the sending.

The function returns the following values:

Return value	Meaning
0	Function completed successfully
-1	CAN task is not ready
-3	Wrong CAN number
-4	The function is not enabled
-5	The sending of Sync messages was released by a different task.
-6	The send buffer of the CAN controller has been assigned a different COB ID.

8.8.5.12 CopyChannelDisplayData (only ETCxC)

Declaration

```
FUNCTION CopyChannelDisplayData: BOOL    (* updates the channel-dependent data of the
                                            NCR in the transferred structure *)
VAR_INPUT
    kanal_b     : BYTE;
    data_pst    : POINTER TO
                  CHANNELDATA_TR;
END_VAR
```

Parameters

kanal_b	Selection of the NC channel (0/1)
data_pst	Pointer to a variable of type CHANNELDATA_TR

Description

The function updates the values of the variables within the structure.

8.8.5.13 CopyCyclicDisplayData (only ETCxC)

Declaration

```
FUNCTION CopyCyclicDisplayData: BOOL    (*updates the data in the transferred structure*)
VAR_INPUT
    data_pst    : POINTER TO CYCLICDATA_TR;
END_VAR
```

Parameters

data_pst	Pointer to a variable of type CYCLICDATA_TR

Description

The function updates the values of the variables within the structure.

8.8.5.14 CopyDiagDisplayData (only ETCxC)

Declaration

```
FUNCTION CopyDiagDisplayData: BOOL    (* updates the data of the structure *)
VAR_INPUT
    data_pst    : POINTER TO DIAGDATA_TR;
END_VAR
```

Parameters

data_pst	Pointer to a variable of type CYCLICDATA_TR

Description

The function updates the values of the variables within the structure.

8.8.5.15 Global constants for CANopen functions

These constants are intended as transfer parameters ”DataType” for the functions CopReadObject() and CopWriteObject(). The values of the constants comply with CiA DS301 ”Application Layer and Communication Profile”.

Constant	Value	Meaning
COP_BOOLEAN_KW	16#1	
COP_INTEGER8_KW	16#2	
COP_INTEGER16_KW	16#3	
COP_INTEGER32_KW	16#4	
COP_UNSIGNED8_KW	16#5	
COP_UNSIGNED16_KW	16#6	
COP_UNSIGNED32_KW	16#7	
COP_REAL32_KW	16#8	
COP_VISIBLE_STRING_KW	16#9	
COP_OCTET_STRING_KW	16#A	
COP_UNICODE_STRING_KW	16#B	
COP_BIT_STRING_KW	16#E	
COP_REAL64_KW	16#11	
Definitions for system variable db0_can_errorstatus_adw		
CAN_TXOVERRUN_KDW	16#20000	Send queue is full
CAN_OVERRUN_KDW	16#10000	Overrun during reception
CAN_BITERR_KDW	16#C000	Transmit bit error
CAN_ACKERR_KDW	16#2000	Acknowledge error
CAN_CRCERR_KDW	16#1000	CRC error
CAN_FORMERR_KDW	16#0800	Message format error
CAN_STUFERR_KDW	16#0200	Transmit error counter >= 96
CAN_RXWARN_KDW	16#0100	Receive error counter >= 96
CAN_IDLE_KDW	16#0040	No error
CAN_FCS_KDW	16#0030	Fault confinement state
CAN_BOFFINT_KDW	16#0004	Bus Off
CAN_ERRINT_KDW	16#0002	Error interrupt
CAN_WAKEINT_KDW	16#0001	No error

8.8.6 CAN functions (only ETCxM)

8.8.6.1 DefineCanMsg (only ETCxM)

Declaration

```
FUNCTION DefineCanMsg: DINT
VAR_INPUT
    CanNum_w      : WORD;    (* Number of the CAN interface - 1 or 2 *)
    WriteCobId_w  : WORD;    (* CobId: ETCxM → CAN node *)
    ReadCobId_w   : WORD;    (* CobId: CAN node → ETCxM *)
    QueueSize_w   : WORD;    (* Number of messages being buffered *)
END_VAR
```

Description

Creating and initialising a structure for managing CAN transfers. This allows to receive or send any CAN messages. When the function is called it checks whether the stated CobId are not still in use. It is also possible for the CobId to be entered into the control configuration of CoDeSys at a CANopen module. ReadCobId_w or WriteCobId_w equals 0 means that no CobId is required yet.

The function returns a handle on the administrative structure. If it returns 0 no administrative structure could be created using the stated parameters.

Example

```
handle_pr       : DINT;

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
```

8.8.6.2 AddCobIdCanMsg (only ETCxM)

Declaration

```
FUNCTION AddCobIdCanMsg: BOOL
VAR_INPUT
    handle_pr     : DINT;    (* Handle on the management structure *)
    CobId_w       : WORD;    (* CobId *)
    Read_bit      : BOOL;    (* direction, TRUE for CAN nodes ETCxM *)
END_VAR
```

Description

This function allows adding a CobId to the management structure (see function DefineCanMsg()) for CAN transfers. The function returns TRUE if the CobId has been entered successfully in the management structure. If the CobId is already in use or the maximum number (32) of possible CobId has been reached, it returns FALSE.

Example

```
handle_pr       : DINT;
FctReturn_bit   : BOOL;

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
FctReturn_bit = AddCobIdCanMsg(handle_pr, 1024, 1);
```

8.8.6.3 DelCobIdCanMsg (only ETCxM)

Declaration

```
FUNCTION DelCobIdCanMsg: BOOL
VAR_INPUT
    handle_pr       : DINT;          (* Handle on the management structure *)
    CobId_w         : WORD;          (* CobId *)
END_VAR
```

Description

This function removes a CobId from the management structure (see function DefineCanMsg()) for CAN transfers. The return value of the function is of no consequence.

Example

```
handle_pr           : DINT;

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
DelCobIdCanMsg(handle_pr, 1014);
```

8.8.6.4 CloseCanMsg (only ETCxM)

Declaration

```
FUNCTION CloseCanMsg: BOOL
VAR_INPUT
    handle_pr       : DINT;          (* Handle on the management structure *)
END_VAR
```

Description

This function deletes the administrative structure for CAN transfers (see function Define CanMsg()= and the memory will be released again. The return value of the function is of no consequence.

Example

```
handle_pr

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
CloseCanMsg(handle_pr);
```

8.8.6.5 ClearCanMsg (only ETCxM)

Declaration

```
FUNCTION ClearCanMsg: BOOL
VAR_INPUT
    handle_pr       : DINT;          (* Handle on the management structure *)
END_VAR
FUNCTION
```

Description

This function can be used to delete the reception FIFO (see function DefineCanMsg()). The return value of the function is of no consequence.

Example

```
handle_pr       : DINT;

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
ClearCanMsg(handle_pr);
```

8.8.6.6 ReadCanMsg (only ETCxM)

Declaration

```
FUNCTION ReadCanMsg: INT
VAR_INPUT
    handle_pr       : DINT;          (* Handle on the management structure *)
    dest_pr         : DINT;          (* Address to the data structure *)
END_VAR
```

Description

With this function a CAN message can be read from the reception buffer. The reception buffer is implemented as FIFO. The structure for reception data is as follows:

```
TYPE CAN_MSG_TR
STRUCT
    cobid_w         : WORD;
    size_b          : BYTE;
    rtr_bit         : BOOL;
    data_ab         : ARRAY [0..7] OF BYTE;
END_STRUCT
END_TYPE
```

The variable dest_pr must always be given the address of a variable of type CAN_MSG_TR (see example). This function will provide an immediate return in any case. The meaning of the return values can be found in the following table.

Return value	Meaning
-2	Invalid handle
0	No new message in FIFO
1	New message available, no further message in FIFO
2	New message available and further messages in FIFO

Example

```
handle_pr       : DINT;
received_bit    : BOOL;
msg_st          : CAN_MSG_TR;

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
handle_pr = DefineCanMsg(1, 1014, 1114, 32);
```

8.8.6.7 WriteCanMsg (only ETCxM)

Declaration

```
FUNCTION WriteCanMsg: INT
VAR_INPUT
    handle_pr   : DINT;     (* Handle on the management structure *)
    dest_pr     : DINT;     (* Address to the data structure *)
END_VAR
```

Description

This function allows for a CAN message to be written into the send buffer. The send buffer is implemented as FIFO. The data structure is as follows:

```
TYPE CAN_MSG_TR
STRUCT
    cobid_w     : WORD;
    size_b      : BYTE;
    rtr_bit     : BOOL;
    data_from   : ARRAY [0..7] OF BYTE;
END_STRUCT
END_TYPE
```

The variables source_pr must always be given the address of a variable of type CAN_MSG_TR (see example). This function will provide an immediate return in any case, even if the FIFO is full. The meaning of the return values can be found in the following table.

Return value	Meaning
-3	Undefined CobId in data structure
-2	Invalid handle or no memory location available for message
-1	FIFO full
0	Message entered into FIFO

Example

```
handle_pr       : DINT;
ret_bit         : BOOL;
src_st          : CAN_MSG_TR;

handle_pr = DefineCanMsg(1, 1014, 1114, 32);
src_st.cobid_w := 1014;
src_st.size_b := 1;
src_st.rtr_bit := FALSE;
src_st.data_from[0] := 27;
ret_bit = WriteCanMsg(handle_pr, ADR(src_st));
```

8.8.7 MMI communication functions

8.8.7.1 GetApplicationMessage

Declaration

```
FUNCTION_BLOCK GetApplicationMessage   (* fetches a message with up to 512 Byte user
                                          data from the MMI *)
VAR_INPUT
    data_pab      : POINTER TO ARRAY[0..511] OF BYTE;
    Datatypes_s   : STRING(255);
END_VAR
VAR_OUTPUT
    sb1_b         : BYTE;
    sb2_b         : BYTE;
    index_b       : BYTE;
    handle_b      : BYTE;
    len_w         : WORD;
    ok            : BOOL;
END_VAR
```

Parameters

data_pab	Pointer to received user data
Datatypes_s	Optional data descriptor, see DefDataTypes
sb1_b	Control block 1 of the message from HMI
sb2_b	Control block 2, optional
index_b	Block counter, optional
handle_b	Sender ID, reserved
len_w	Length of received user data in bytes
ok	TRUE, if a message has been received

Description

With this function the PLC can fetch a message of type SB0_SPSAUFTRAG_KUC (SB0=14) with up to 512 Byte user data from HMI. Here the message buffer between NCR and MMI in the dual port RAM will be used. The function can be used together with PutApplicationMessage to establish a fast message communication between the PLC and HMI.

In *data_pab* a pointer to a buffer can be transferred where the received message should be stored. For this it must be ensured that the buffer is large enough to accept the received message. When starting or resetting the PLC this pointer will automatically be initialised to the start of DB7 (%MW7.0).

If the format of the message to be received is known in advance, the data descriptor *datatypes_s* can be used to automatically carry out the possibly required byte swapping. Otherwise this must be carried out later with MOVESWAPPED or the functions GET_WORD, GET_DWORD, etc. have to be used to fetch the data from the message. If the defined descriptor string is faulty the data will be transferred without byte swapping.

If a message from HMI is available when calling GetApplicationMessage, OK is set to TRUE and the message copied into the defined buffer. The function provides an immediate return in any case, even if no message from HMI is available. In that case **OK** is set to **FALSE**.

By evaluating *sb1_b* the message can be identified by the PLC. The definition of control blocks must take place between the PLC and MMI developer. The length of the user data is stored in *len_w* in bytes.

The variables *sb2_b* and *index_b* are optional. Their use must be resolved between the PLC and HMI developer. *handle_b* is reserved for the simultaneous communication with several applications.

Note!

These messages are always transferred unsynchronised. Up to 10 messages of this type form MMI to PLC can be buffered in the control, the rest will be lost. In this case the control reports a corresponding error message. If necessary, synchronisation mechanisms must be agreed between HMI and PLC.

8.8.7.2 PutApplicationMessage

Declaration

```
FUNCTION_BLOCK PutApplicationMessage   (* sends a message with up to 512 Byte user
                                          data to the MMI *)
VAR_INPUT
    data_pab      : POINTER TO ARRAY[0..511] OF BYTE;
    Datatypes_s   : STRING(255);
    sb1_b         : BYTE;
    sb2_b         : BYTE;
    index_b       : BYTE;
    handle_b      : BYTE;
    len_w         : WORD;
END_VAR
VAR_INPUT
    ok            : BOOL;
END_VAR
```

Parameters

data_pab	Pointer to user data to be sent
Datatypes_s	Optional data descriptor, see DefDataTypes
sb1_b	Control block 1 of the message to HMI
sb2_b	Control block 2, optional
index_b	Block counter, optional
handle_b	Receiver ID, reserved
len_w	Length of received user data in bytes
ok	TRUE, if a message has been sent

Description

With this function the PLC can send a message of type SB0_SPSAUFTRAG_KUC (SB0=14) with up to 512 Byte user data to HMI. Here the message buffer between NCR and MMI in the dual port RAM will be used. The function can be used together with GetApplicationMessage to establish a fast message communication between the PLC and MMI.

In *data_pab* a pointer to a buffer containing the user data to be sent can be transferred. When starting or resetting the PLC this pointer is automatically initialised to the beginning of DB6 (%MW6.0).

The data descriptor *datatypes_s* can be used to automatically carry out the possibly required byte swapping of the user data. Otherwise this must first be carried out using MOVESWAPPED or the functions PUT_WORD, PUT_DWORD, etc. must be used to write data to the message.

If *datatypes_s* contains an empty string the number of the user data bytes to be sent must be stated in *len_w*. Otherwise the lengths will be provided by the data descriptor.

sb1_b defines the type of message. For every user data format used a separate control block should be used. The definition must be agreed between the PLC and MMI developer.

The variables *sb2_b* and *index_b* are optional. Their use must be resolved between the PLC and HMI developer. *handle_*b is reserved for the simultaneous communication with several applications.

After entering the message into the send buffer to the HMI OK will be set to TRUE. In case of a faulty descriptor string *datatypes_s* or a send buffer overrun OK will be set to FALSE. The function will immediately provide a return even if a message could not be sent. In the case of a send buffer overrun it can, of course, be called again later.

Note!

These messages are always transferred unsynchronised. Up to 30 messages of this type from NCR to MMI can be buffered in the control in total, the rest will be lost. In this case the control reports a corresponding error message. If necessary, synchronisation mechanisms must be agreed between HMI and PLC.

8.8.8 Realtime clock (only ETCxM)

8.8.8.1 RTC_GetTime_DT (only ETCxM)

Declaration

```
FUNCTION RTC_GetTime_DT: BOOL
VAR_INPUT
    GetTime_pr      : DWORD;
END_VAR
```

Description

This function allows for the realtime clock of the control to be read. The function must be given the address of a variable of type DT (see example). The return value TRUE of the function indicates that the time could be read.

Example

```
ret_bit             : BOOL;
time_dt             : DT;

ret_bit = RTC_GetTime_DT(ADR(time_dt));
```

8.8.8.2 RTC_SetTime_DT (only ETCxM)

Declaration

```
FUNCTION RTC_SetTime_DT: BOOL
VAR_INPUT
    SetTime_dt      : DT;
END_VAR
```

Description

With this function the realtime clock of the control is set to the transferred value. The return value TRUE of the function indicates that the time has been written to the clock component.

Example

```
ret_bit             : BOOL;
time_dt             : DT := dt#2003-11-15-00:00:00;

ret_bit = RTC_SetTime_DT(time_dt);
```

8.9 Library ServerSDO.lib

8.9.1 InitServerSdo

Declaration

```
FUNCTION InitServerSdo: BOOL
VAR_INPUT
    Max_NetVarODIdx     : UINT;
    pNetVarOD           : POINTER TO NetVarOD_CAN;
END_VAR
```

Parameters

Max_NetVarODIdx	maximum index
pNetVarOD	Pointer to the object directory

Description

This function announces an object directory for the data transfer with a CANopen client to the runtime system.

Example

To enable the CoDeSys object directory the following function must be called once in the PLC program:

```
InitServerSdo(MAX_NetVarOD_CAN, ADR(pNetVarOD_CAN[0]));
```

9 Index

A

AddCobIdCanMsg, 410

address CAN, 58

Addressing
- Data block, 350
- I/O module, 349

Addressing , 349

ALLOCV24, 377

Appendix, 321

application, as intended, 12

application as directed, 12

Arithmetic operations, 172

Automation system, Example, 13

B

Block extensions, 174

Block preprocessing, 90

boot monitor, call, 83

Bootloader, 363

Bus termination
- CAN bus, 15
- ME bus, 15

C

CAN
- address, 58
- Bus termination, 15
- Configuring Master, 344
- Searching modules, 348
- Slave configuration, 345

CE conformity, 12

CenterString, 361

Check drives, with ETC-MMI, 43 , 44

Checking IO number, 347

ClearCanMsg, 412

CloseCanMsg, 411

CLRRXBUFFER, 380

CLRTXBUFFER, 381

CNC program
- enter, 48
- error message, 74
- extend, 50
- load, 49
- save, 48
- start, 49
- test, 72

CNC programming, 45

CoDeSys, 337 , 340 , 362
- Addressing, I/O modules, 349
- Addressing , 349
  Data blocks, 350
- CAN slave configuration, 345
- Checking IO number, 347
- configure control system, 51
- Configuring CAN Master , 344
- Configuring I/O modules , 343
- Configuring PLC tasks, 340 , 342
- create project, 52
- Data blocks , 356
- Error messages, 360
- Functions
  AddCobIdCanMsg, 410
  ALLOCV24, 377
  CANopen functions, 409
  CenterString, 361
  ClearCanMsg, 412
  CloseCanMsg, 411
  CLRRXBUFFER, 380
  CLRTXBUFFER, 381
  CopDefineDS403, 398
  CopGetNodeState, 400
  CopReadObjekt, 401
  CopWriteObject, 403
  CopXDefineDS403, 400
  CopXDisableSync, 407
  CopXEnableSync, 407
  CopXReadObject, 402
  CopXSetOperational, 405
  CopXSetPreoperational , 406
  CopXWriteObject, 404
  CopyChannelDisplayData, 408
  CopyCyclicDisplayData, 408
  CopyDiagDisplayData, 408
  DEFDATATYPES, 393
  DefineCanMsg, 410
  DelCobIdCanMsg, 411
  Device driver, 390
  DrvReadObject, 401
  DrvWriteObject, 403
  FMOD, 361
  Format, 362
  FREEV24, 378

general, 361
GET_BYTE, 395
GET_DINT, 395
GET_INT, 395
GET_LREAL, 395
GET_REAL, 395
GET_WORD, 395
GetApplicationMessage, 414
GetFirmwareVersion, 363
GetMacAddr, 364
GetUserParam, 364
INITV24, 378
IO_SET, 365
LOAD, 376 , 383
LOAD_PARAM, 376
Load_Param, 365
MEMCOMP, 395
MEMCOPY, 395
MEMSET, 396
OVESWAPPED, 396
PUT_BYTE, 397
PUT_DINT, 397
PUT_DWORD, 397
PUT_INT, 397
PUT_LREAL, 397
PUT_WORD, 397
PutApplicationMessage, 415
READ_PARAM_DINT, 366
READ_PARAM_INT, 365
READ_PARAM_LREAL, 366
READ_PARAM_REAL, 366
READ_SYSPARAM, 366
READ_TOOLDATA, 367
READBLOCKV24, 379
ReadCanMsg, 412
READV24, 379
RTC_GetTime_DT, 417
RTC_SetTime_DT, 417
SAVE, 376 , 383
SAVE_PARAM, 368 , 376
SetCurrentPath, 384
SETINPUT_BIT, 368
SETINPUT_WORD, 369
SETLANGUAGE, 372
SINGLEBLOCK, 369
SPSERROR, 370
STRTOF, 372
STRTOL10, 372
SYSCLOSEFILE, 385
SYSDISKFORMAT, 387
SYSDISKINFO, 389
SYSERROR, 373
SYSFIRSTFILE, 388
SYSNEXTFILE, 388
SYSOPENFILE, 384
SYSREADFILE, 385
SYSREADLINE, 386
SYSREMOVEFILE, 387
SYSWRITEFILE, 386
TRACE, 373
V24 function constants, 382 , 391
WordWrap, 374
WRITE_PARAM_DINT, 374
WRITE_PARAM_INT, 374
WRITE_PARAM_LREAL, 375
WRITE_PARAM_REAL, 375
WRITE_SYSPARAM, 375
WRITEBLOCKV24, 379
WriteCanMsg, 413
WRITEV24, 380
- installation, 337
- Interface to the ETC, 356
- library, 361
- Library , ServerSDO.lib, 418
- Machine constants, 359
- Network variables, 353
- Object directory, 350
- Operating data, 360
- Parameter field, 359
- Parameter Manager, 350
- programming with, 51 , 336
- Remanent variables, 350
- Searching CANopen modules, 348
- start, 52
- System variables, 356 , 359
- Target system settings, 353
- Target system setup, 340
- Variable list, global, 354 , 355

Commissioning, commissioning steps, overview, 20

Configure control system, 51

Configure I/O modules, 65

Configuring I/O modules , 343

Configuring PLC tasks, 340 , 342

conformity, 12

Connecting ETC and PC, 338

Connecting PC and ETC, 338

Constants
- CANopen functions, 409
- I/O functions, 391
- V24 functions, 382

CopDefineDS403, 398

CopGetNodeState, 400

CopReadObjekt, 401

CopWriteObject, 403

CopXDefineDS403, 400

CopXDisableSync, 407

CopXEnableSync, 407

CopXReadObject, 402

CopXSetOperational, 405

CopXSetPreoperational , 406

CopXWriteObject, 404

CopyChannelDisplayData, 408

CopyCyclicDisplayData, 408

CopyDiagDisplayData, 408

cycle programming, 310 , 325

D

Data block, 63

Data block 0, 242

Data block 1, 245

Data block 15, 273

Data block 2, 268

Data block 8 ... 14, 270

Data blocks , 356

data fields
- P field, 180
- Q field, 189

DEFDATATYPES, 393

Define inputs/outputs, 65

DefineCanMsg, 410

DelCobIdCanMsg, 411

Device driver, 390

DPR interface, 339

drive controller
- drive controller, 12
- marking, 12

Drives, parameterise, 36

DrvReadObject, 401

DrvWriteObject, 403

E

Editor, 47

editor, 305

EDS file, 55

Error messages, 360

ETC interface PLC-NC
- Data block 0, 242
- Data block 1, 245
- Data block 15, 273
- Data block 2, 268
- Data block 8 ... 14, 270

ETC island, 15

ETC programming
- Arithmetic operations, 172
- Block extensions, 174
- Block preprocessing, 90
- data fields, 180 , 189
- Formula processor , 172
- G functions, 91
- G preparatory function, 89
- H functions, 175
- M functions, 175
- Modal functions, 89
- Parameter assignment P, 174
- Program components, 87
- Program creation variants, 87
- Program transfer, 87
- Q functions, 177
- S functions, 178 , 179
- Time synchronization, 90

ETC-MMI
- adapt machine constants, 42
- check drives, 43 , 44
- check machine constants, 43 , 44
- configuration file, 327
- configure, 30
- cycle programming, 310 , 325
- DELPHMMI.INI, 327
- description, 286
- dialogue box for file selection, 306
- editor, 47 , 305
- establish connection, 34
- file manager, 308
- help function, 291
- install, 31 , 286
- language file, 321
- messages, 292
- operate, 289 , 290
- operating mode ”Automatic”, 298
- operating mode ”diagnostics”, 312
- operating mode ”Programming”, 302
- operating mode ”setup”, 293
- password, 292
- PLC keys, 75
- split screen, 289
- start, 32 , 288
- switch language, 33

ETC-MMI gateway
- communication values in the DPR area, 285
- configuration interface, 278
- configure, 30 , 278
- configured connections, 278
- description, 276
- error logbook, 281 , 283
- establish connection, 34
- install, 276

- Mmigtway.ini, 283
- operating mode , 277
- start, 277
- Version information, 282

ETCHx
- connect with PC, 16
- Connections, 15
- Description, 13
- IP address, 28
- operating mode, 25
- start, 21

ETCPx
- Description, 13
- install, 31

ETCxC.mk, 39

Ethernet interface, 338

Example
- Automation system, 13
- PLC program, 60

example, CNC program, 47

F

Fault elimination, monitor interface, 24

File
- EDS, 55
- ETCxC.mk, 39

file, DelphMMI.ini, 327

file manager, 308

Firmware, update, 83

FMOD, 361

Format, 362

Formula processor , 172

FREEV24, 378

Functions
- AddCobIdCanMsg, 410
- ALLOCV24, 377
- CANopen functions, 409
- CenterString, 361
- ClearCanMsg, 412
- CloseCanMsg, 411
- CLRRXBUFFER, 380
- CLRTXBUFFER, 381
- CopDefineDS403, 398
- CopGetNodeState, 400
- CopReadObjekt, 401
- CopWriteObject, 403
- CopXDefineDS403, 400
- CopXDisableSync, 407
- CopXEnableSync, 407
- CopXReadObject, 402
- CopXSetOperational, 405
- CopXSetPreoperational , 406
- CopXWriteObject, 404
- CopyChannelDisplayData, 408
- CopyCyclicDisplayData, 408
- CopyDiagDisplayData, 408
- DEFDATATYPES, 393
- DefineCanMsg, 410
- DelCobIdCanMsg, 411
- Device driver, 390
- DrvReadObject, 401
- DrvWriteObject, 403
- FMOD, 361
- Format, 362
- FREEV24, 378
- general, 361
- GET_BYTE, 395
- GET_DINT, 395
- GET_INT, 395
- GET_LREAL, 395
- GET_REAL, 395
- GET_WORD, 395
- GetApplicationMessage, 414
- GetFirmwareVersion, 363
- GetMacAddr, 364
- GetUserParam, 364
- I/O functions, 391
- INITV24, 378
- IO_SET, 365
- LOAD, 376 , 383
- LOAD_PARAM, 376
- Load_Param, 365
- MEMCOMP, 395
- MEMCOPY, 395
- MEMSET, 396
- OVESWAPPED, 396
- PUT_BYTE, 397
- PUT_DINT, 397
- PUT_DWORD, 397
- PUT_INT, 397
- PUT_REAL, 397
- PUT_WORD, 397
- PutApplicationMessage, 415
- READ_PARAM_DINT, 366
- READ_PARAM_INT, 365
- READ_PARAM_LREAL, 366
- READ_PARAM_REAL, 366
- READ_SYSPARAM, 366

- READ_TOOLDATA, 367
- READBLOCKV24, 379
- ReadCanMsg, 412
- READV24, 379
- RTC_GetTime_DT, 417
- RTC_SetTime_DT, 417
- SAVE, 376 , 383
- SAVE_PARAM, 368 , 376
- SetCurrentPath, 384
- SETINPUT_BIT, 368
- SETINPUT_WORD, 369
- SETLANGUAGE, 372
- SINGLEBLOCK, 369
- SPSERROR, 370
- STRTOF, 372
- STRTOL10, 372
- SYSCLOSEFILE, 385
- SYSDISKFORMAT, 387
- SYSDISKINFO, 389
- SYSERROR, 373
- SYSFIRSTFILE, 388
- SYSNEXTFILE, 388
- SYSOPENFILE, 384
- SYSREADFILE, 385
- SYSREADLINE, 386
- SYSREMOVEFILE, 387
- SYSWRITEFILE, 386
- TRACE, 373
- V24 function constants, 382
- WordWrap, 374
- WRITE_PARAM_DINT, 374
- WRITE_PARAM_INT, 374
- WRITE_PARAM_LREAL, 375
- WRITE_PARAM_REAL, 375
- WRITE_SYSPARAM, 375
- WRITEBLOCKV24, 379
- WriteCanMsg, 413
- WRITEV24, 380

G

G function
- Individual descriptions, 94
- overview, 91

G preparatory function, 89

G-function, basics, 45

G00 Point-to-point positioning, high rate, 94

G01 Linear interpolation, 97

G02 Circular interpolation, 99

G04 Dwell time, 102

G05 Spline interpolation, 103

G06 Polynomial interpolation, 105

G100 Polar coordinates: linear interpolation, high rate, 126

G101 Polar coordinates: linear interpolation, 127

G102/G103 Polar coordinates: circular interpolation, 128

G110 Polar coordinates: accept center, 129

G111 Polar coordinates: center programming, 129

G112 Tangential correction on, 130

G113 Tangential correction off, 131

G114 6-axes transformation, 131

G115 Convex surface transformation, 132

G116 Rotation axes transformation, 133

G120 Abs. coordinate shift via modal offset, 134

G121 Programming the modal offset, 135

G122 Configuring the effect of the traverse keys, 136

G125 Parameter field comparison, 137

G130 Parameter field comparison, 138

G131 Delete modal comparative operation, 140

G132 Inhibit modal program branching, 140

G133 Enable modal program branching, 140

G134 Parameter field comparison, 141

G140/G141/G142 Noncircular grinding, 142

G143 Parameters of grinding phases during noncircular grinding, 144

G144/G145 Programming a correction table during noncircular grinding, 145

G150 Q-Bit comparison, 146

G151 Q-Bit comparison, 148

G152 Q-Bit comparison, 149

G153 Wait for the termination of a channel 1, 150

G158/G159 Intermittent operation, 151

G16 Selection of the principal and secondary axes, 107

G161 Accept actual position, 152

G162 Define axis group, 153

G17, G18, G19 Plane selection, 108

G175 Axis replacement, 154

G180 Modal travel , 155

G181 Modal travel , 156

G187 Handwheel, 156

G193 Set absolute zero point, 157

G194 Program an add. basic offset, 157

G195 Absolute coordinate shift , 158

G20 Block jump, 108

G200 Geometry filter , 159

G201 Change the acceleration and deceleration ramps, 160

G209 Set the geometry counter, 161

G211 Transformation for two-axle articulated robot kinematics, 161

G22 Subprogram call, 109

G226 Reconfigure hardware limit switch, 163

G233 2D/3D axis correction, 164

G234 Direction-dependent height correction, 166

G24 Define pos. traversing range limit, 110

G25 Define neg. traversing range limit, 111

G251 Accept step response of an axis, 167

G252 Value input via display device, 168

G253 Output of a comment, 170

G26 Activate traversing range limits, 111

G27 Jump function with repetition counter, 112

G30, G31 Route or path control, 112

G33 Coupling , 113

G34 Coupling , 114

G36 Modal oscillation , 114

G37 Modal oscillation , 115

G40 Deactivate tool correction, 115

G41, G42 Tool path correction, 116

G53 Deactivate temp. coordinate shift, 117

G54 Temporary zero shift, 117

G60 Exact positioning , 118

G61 Stop block preprocessing, 119

G74 Home position approach, 119

G75 Scaling factor for input units, 120

G76 Scaling factors for pulse evaluation, 120

G88 Basic rotation, 121

G89 Profile rotation, 122

G90 Absolute dimensions (reference dimension), 123

G91 Increm. dimensions (incremental dimension progr.), 123

G92 Relative zero shift, 124

G93 Absolute zero shift, 125

G99 Return, 125

Generate program, 355

GET_BYTE, 395

GET_DINT, 395

GET_DWORD, 395

GET_INT, 395

GET_LREAL, 395

GET_REAL, 395

GET_WORD, 395

GetApplicationMessage, 414

GetFirmwareVersion, 363

GetMacAddr, 364

Getting started, 13

GetUserParam, 364

Guard time, 59

H

H functions, 175

HMI, 77

HyperTerminal, 21

I

INITV24, 378

Interface
- DPR, 339
- Ethernet, 338
- monitor, 21 , 24
- V.24, 338

Interface to the ETC, 356

IO_SET, 365

IP address, ETCHx, 28

IPC
- Applications, 14
- Description, 14

L

Language, switch, ETC-MMI, 33

Legal regulations, 12

LENZE-HMI, 77

liability, 12

library, 361

Library , ServerSDO.lib, 418

Life time factor, 59

LOAD, 376 , 383

LOAD_PARAM, 376

Load_Param, 365

M

M functions, 175

M-function, basics, 46

MAC address, 29

Machine constants, 95 , 160 , 359
- adapt file, 41 , 42
- check, 43 , 44
- Configuration of axes, Assignment and evaluation, 207
- Configuration of axes, Basics, 204
- Configuration of axes, Controller settings, 211
- Configuration of axes, Correction of axes, 215
- Configuration of axes, Handwheels, 218
- Configuration of axes, Operating range, 210
- Configuration of axes, Referencing, 212
- Configuration of axes, Resolution, 209
- Configuration of axes, speed and acceleration, 213
- Configuration of axes, Synchronous axes, 220
- ETCxC.mk, 39
- load file into control system, 40
- Overview, 226
- overview, 36
- Software configuration , 193
- Storage space reservation, 200
- Technology-specific settings, 221
- Test settings, 192

machine constants, 112

manufacturer, 12

ME bus termination, 15

MEMCOMP, 395

MEMCOPY, 395

MEMSET, 396

MK_ACHSENART, 208 , 218 , 220

MK_APPLACHSIDX, 205 , 207 , 218

MK_BAHNBESCHL, 214

MK_BAHNBREMS, 214

MK_BESCHL, 213

MK_BREMS, 213

MK_CANDRIVES, 204 , 207 , 218

MK_CANOPEN_BAUDRATE, 198

MK_CONST_REL_INCH, 195

MK_CONST_REL_MM, 195

MK_DELTAT, 199

MK_DW224_255, 225

MK_EPSILONGRAD, 197

MK_EPSILONMM, 197

MK_FEHLERRESTART, 197

MK_GENAUHALTZEIT , 210

MK_GEWINDE_VMAX, 224

MK_GRUNDOFFSET, 210

MK_HANDRADFAKTOR, 219

MK_HANDRADFILTER, 219

MK_HANDRADZUORDNUNG, 218

MK_HEADERANZAHL, 201

MK_IMPULSE, 209 , 218

MK_KANALANZAHL, 201

MK_KARTESISCH_ACHSNR, 222

MK_KONTURFEHLER, 195

MK_KUNDE, 193

MK_LAH_GRENZWINKEL, 196

MK_LAH_RUECKLAUFGRENZE, 202

MK_LAH_VORLAUFTIEFE, 202

MK_MASCH_POLAR_KART, 222

MK_MASSSTAB, 209

MK_METRISCH, 194

MK_MFKT_UPR_TABELLE, 221

MK_MODVMAX, 213

MK_NCPROG_NICHT_INS_EEPROM, 194

MK_NCPROG_OHNE_KOMMENTARE, 194

MK_OVERRIDEMAX, 197

MK_PFELD_GROESSE, 203

MK_POLAR_ACHSNR, 223

MK_RADIUS_B_BEWERTUNG, 196

MK_REF_RICHTUNG_UND_FOLGE, 212

MK_S_VERSATZSPERRE, 198

MK_S0T0_VERSATZ_ERLAUBT, 198

MK_SCHLEPPGENAUHALT, 210

MK_SPINDELUMKEHRSPIEL, 215

MK_SPS_DATENGROESSE, 200

MK_SPS_DUMMY, 192

MK_SPS_SPEICHERGROESSE, 200

MK_SPV_SPEICHERGROESSE, 200

MK_SPV_SYMBOLANZAHL, 201

MK_SW_ENDS_MINUS, 210

MK_SW_ENDS_MIT_RAMPE, 210

MK_SW_ENDS_PLUS, 210

MK_SYNCHRONABWEICHUNG, 220

MK_SYNCHRONOFFSET , 220

MK_T_BAHNBESCHL, 214

MK_T_BESCHL , 213

MK_T2, 211

MK_TECHNOLOGIEDATEN, 221

MK_TEST_OHNEMECHANIK, 192

MK_VBAHNMAX, 214

MK_VMAX, 213

MK_VOREINSTELLUNG, 193

MK_WEG, 209 , 218

MK_WLK_C_GRENZWINKEL, 223

MK_WLK_C_OFFSET, 223

MK_WLK_VERWEILZEIT, 224

MK_X_WINKEL, 224

Mmigtway.ini, 283

Modal functions, 89 , 90

Monitor interface
- activate, 21
- commands, 24
- fault elimination, 24

MUSTER.MK, 226

N

nameplate
- control system ETCHx, 11
- control system ETCPx, 11
- module ETCHx, 11

Network variables, 353

Nodeguarding, 59 , 346 , 348

O

Object directory, 350

Operating data, 360

operating mode MMI
- ”Automatic”, 298
- ”diagnostics”, 312
- ”Programming”, 302
- ”Setup”, 293

Operating modes ETCxH, 25

Overview
- Arithmetic operations, 172
- G function, 91
- Machine constants, 226
- machine constants, 36

OVESWAPPED, 396

P

P field, 180

Parameter
- System, 181
- Technology-specific, 186
- Time recording, 187

Parameter assignment P, 174

Parameter field, 359

Parameter Manager, 350

Parameters
- Axis-specific, 181
- Program management, 183
- Technology-specific user parameters, 187

Paramterise, drives, 36

PC
- Applications, 14
- Description, 14

PLC key, 75

PLC program
- add libraries, 62
- configure, 61
- create, 60
- Example, 60
- load, 68 , 71
- operate, 71
- sequence control, 64
- start, 61 , 71
- test, 72

product identification
- control system ETCHx, 11
- control system ETCPx, 11
- module ETCHx, 11

Programming with CoDeSys, 51 , 336

PUT_BYTE, 397

PUT_DINT, 397

PUT_DWORD, 397

PUT_INT, 397

PUT_LREAL, 397

PUT_REAL, 397

PUT_WORD, 397
PutApplicationMessage, 415

Q

Q field, 189
Q functions, 177

R

READ_PARAM_DINT, 366
READ_PARAM_INT, 365
READ_PARAM_LREAL, 366
READ_PARAM_REAL, 366
READ_SYSPARAM, 366
READ_TOOLDATA, 367
READBLOCKV24, 379
ReadCanMsg, 412
READV24, 379
Remanent variables, 350
RTC_GetTime_DT, 417
RTC_SetTime_DT, 417

S

S functions, 178
SAVE, 376 , 383
SAVE_PARAM, 368 , 376
Sequence control, 64
ServerSDO.lib, 418
SetCurrentPath, 384
SETINPUT_BIT, 368
SETINPUT_WORD, 369
SETLANGUAGE, 372
SINGLEBLOCK, 369
SPRACHE.TXT, 321
SPSERROR, 370
Status message, 17
STRTOF, 372
STRTOL10, 372
SYSCLOSEFILE, 385
SYSDISKFORMAT, 387
SYSDISKINFO, 389
SYSERROR, 373
SYSFIRSTFILE, 388
SYSNEXTFILE, 388
SYSOPENFILE, 384
SYSREADFILE, 385
SYSREADLINE, 386
SYSREMOVEFILE, 387
System variables, 63 , 356 , 359
SYSWRITEFILE, 386

T

T functions, 179
Target system settings, 353
Target system setup, 340
Terminal program, 21
Test
- CNC program, 72
- PLC program, 72

TRACE, 373

U

Update, firmware, 83

V

V.24 interface, 338
Variable list, global, 354

W

warranty, 12
WordWrap, 374
WRITE_PARAM_DINT, 374
WRITE_PARAM_INT, 374
WRITE_PARAM_LREAL, 375
WRITE_PARAM_REAL, 375
WRITE_SYSPARAM, 375
WRITEBLOCKV24, 379
WriteCanMsg, 413
WRITEV24, 380

Lenze Drive Systems GmbH
Hans-Lenze-Straße 1
D-31855 Aerzen
Germany

☎ +49 (0) 51 54 82-0
☎ Service 00 80 00 24 4 68 77 (24 h helpline)
🖷 Service +49 (0) 51 54 82-1112
E-Mail Lenze@Lenze.de
Internet www.Lenze.com

EDSTCXN
EN 2.0
© 08/2006
TD29

10 9 8 7 6 5 4 3 2 1